KITTU PATEL

VERICORE

1000+ INTERVIEW QUE

SYSTEM VERILOG

1000 Must-Know SystemVerilog Interview Questions

VeriCore Exclusive Resource

Section 1: Constraint Logic & Randomization (Q1–25)

1. **What happens if you use an inside constraint with an empty set? What are the implications?**
2. **How does SystemVerilog solve conflicting constraints? Explain with an example.**
3. **Can ‘rand‘ and ‘randc‘ be used together in the same object? Justify.**
4. **What is the difference between ‘rand‘ and ‘randc‘ and in what practical cases would you prefer one over the other?**
5. **Write a constraint to generate a 4-digit even number where all digits are unique.**
6. **If a constraint has multiple solutions, how does SystemVerilog pick one? Explain.**
7. **Can constraint solver fail silently? How can you detect such failures during simulation?**
8. **Design a class with ‘rand‘ members and write inline constraints to limit values between 100 and 200 that are divisible by 4 but not by 8.**
9. **What is the difference between soft constraints and normal constraints? Give real-world use cases.**
10. **Is it possible to randomize an object without any constraints being applied? How?**
11. **What is constraint inheritance? How does it behave when constraints are redefined in a derived class?**
12. **Create a constraint to ensure an array of size 5 contains only prime numbers.**
13. **Why might a constraint block be declared as static? What does that imply?**

14. **Explain ‘solve before‘ constraint directive with example. What problem does it solve?**

15. **When is ‘randcase‘ preferred over regular constraints? Can you mix them?**

16. ‘XX:XX:XX:YY:YY:YY‘, where X and Y are constrained groups.**Design a class to generate MAC addresses in a specific pattern:**

    ‘XX:XX:XX:YY:YY:YY‘, where X and Y are constrained groups.

17. **What are the side effects of over-constraining a class? How do you detect and fix this?**

18. **Can constraints be turned off selectively? Explain the use of constraint mode control.**

19. **Give an example where randomization fails due to cross-product of conditions.**

20. **Design a constraint that randomizes a list of 10 numbers such that the sum is always divisible by 5.**

21. **What are post-randomize functions? How are they used in verification environments?**

22. **How would you test the uniformity of a constraint-based randomized value in a testbench?**

23. **Can class handles be randomized? If yes, what gets randomized and what doesn’t?**

24. **Explain how SystemVerilog solves hierarchical constraints.**

25. **What is the difference between constraint solver and regular procedural code logic? Why can’t we just use ‘if‘ statements inside constraints?**

Section 2: Object-Oriented Programming (Q26–50)

26. **What is the difference between ‘new()‘ constructor and factory pattern in SystemVerilog? When should each be used?**

27. **Explain the significance of ‘virtual‘ keyword in class declarations. What happens if it is not used?**

28. **Can a virtual method be overridden with a non-virtual method in a derived class? Justify your answer.**

29. **Write a SystemVerilog example demonstrating dynamic polymorphism using virtual classes.**

30. **What are pure virtual methods? When are they necessary? Provide a real-time UVM scenario where they are used.**

31. **Describe the concept of late binding in SystemVerilog OOP. How does it affect simulation?**

32. **What is the significance of the ‘super‘ keyword? Explain with an example.**

33. **How does SystemVerilog handle object slicing? Is it relevant like in C++?**

34. **Design a base and two derived classes with overridden methods. Call the methods polymorphically.**

35. **What are the main differences between static and dynamic class properties? Give verification-specific examples.**

36. **Can we overload constructors in SystemVerilog? Show multiple constructor definitions with use-cases.**

37. **How can you achieve abstraction in SystemVerilog OOP? Illustrate with a testbench design.**

38. **Compare deep copy and shallow copy in context of ‘copy()‘ method in class handles.**

39. **What is the role of ‘extern‘ methods in class-based designs? How does it improve readability?**

40. **Demonstrate encapsulation in SystemVerilog using ‘local‘ and ‘protected‘ keywords.**

41. **How do ‘typedef class‘ and ‘typedef enum‘ enhance modularity in SystemVerilog OOP design?**

42. **What are the best practices while using inheritance in verification environments? List at least 5.**

43. **What happens if a child class does not override a pure virtual method? What error is thrown?**

44. **Explain parameterized classes with an example where width and depth are parameters.**

45. **Why is handle-based object management essential in UVM? Relate to class inheritance.**

46. **Can we use interfaces inside classes? Explain interaction between OOP and RTL boundaries.**

47. **Write a class that generates a sequence item with multiple overrides based on the DUT mode.**

48. **What is the lifecycle of a class object in SystemVerilog? How is memory allocation managed?**

49. **Explain the use of abstract classes and polymorphism in building layered testbenches.**

Section 3: Constraint-Based Code Problems (Q51–75)

51. **Write a class to randomize a 6-digit number such that the sum of all digits is always 30.**

52. **Create a constraint to randomize a byte where all bits are alternating (i.e., 10101010 or 01010101).**

53. **Write a constraint to randomize two variables ‘a‘ and ‘b‘ such that ‘a ¿ b‘, ‘a + b = 50‘, and both are even.**

54. **Randomize a 4-element array such that the difference between each adjacent element is exactly 3.**

55. **Design a constraint to randomize 5 integers such that they form a valid arithmetic progression.**

56. **Randomize an array of size 6 with values from 1 to 9 such that no value repeats and the sum is divisible by 3.**

57. **Create a constraint that generates a random 8-bit number where the Hamming weight is exactly 4.**

58. **Randomize two variables ‘x‘ and ‘y‘ such that ‘x*y = 100‘ and both ‘x‘ and ‘y‘ are positive integers less than 50.**

59. **Write a class to randomize an IP address of the format ‘A.B.C.D‘ where A must be in [192-223], B and C are any 8-bit values, and D is not 0 or 255.**

60. **Randomize a time value in HH:MM:SS 24-hour format using constraints. Ensure all fields are valid.**

61. **Design a constraint to generate a 5-digit palindrome number.**

62. **Create a constraint to randomize a binary string of length 10 such that it starts and ends with ’1’.**

63. **Randomize a password of length 6 using characters from ‘A-Z, a-z, 0-9‘ with at least one uppercase letter.**

64. **Randomize a variable ‘count‘ such that it is divisible by both 3 and 5, but not by 2, and lies between 30 and 150.**

65. **Write a constraint that generates a random date in the format ‘DD-MM-YYYY‘ ensuring leap year handling.**

66. **Design a constraint to generate a set of coordinates ‘(x, y)‘ such that they lie within a circular region of radius 5.**

67. **Randomize an array of 10 integers such that no two consecutive elements are equal.**

68. **Create a constraint to generate a random boolean truth table (8 bits) ensuring only one '1' appears.**

69. **Write a class that generates a sequence of increasing numbers randomly between 0 and 50.**

70. **Create a constraint to randomize a 4-letter string where all characters are vowels.**

71. **Randomize a phone number pattern: '+91-XXXXX-XXXXX' ensuring first digit after +91 is non-zero.**

72. **Generate a constraint-based Sudoku row with numbers 1–9 (no repetitions).**

73. **Write a constraint to ensure that the XOR of 4 randomly generated 8-bit numbers is always zero.**

74. **Create a constraint to generate an array where sum of even-indexed elements equals sum of odd-indexed elements.**

75. **Design a class that generates random values for 'age' and 'category' where:**

    - **if age < 18 ⇒ category = 'minor'**
    - **if age between 18–60 ⇒ category = 'adult'**
    - **if age > 60 ⇒ category = 'senior'**

    Use conditional constraints.

Section 4: Constraint Solver Edge Cases + Output Prediction (Q76–100)

76. **Predict the output:**

```
class test;
  rand bit [3:0] a;
  constraint c1 { a > 5; a < 4; }
endclass
```

What will happen when we call 'randomize()' on the object of this class? Why?

77. **Explain what happens when a constraint contradicts a 'pre**$_r$*andomize*()'*valuemanuallysetby*

78. **What is the output of the following and explain why?**

```
class test;
  rand bit [2:0] a;
  constraint c1 { a inside {[1:3]}; }
endclass
```

```
initial begin
  test t = new();
  repeat (5) begin
    t.randomize();
    $display("a = %0d", t.a);
  end
end
```

79. **What happens when no ‘rand‘ keyword is used but constraints are defined? Will the solver execute?**

80. **Output Prediction:**

```
class test;
  rand bit [3:0] a, b;
  constraint c { a == b; a > 10; b < 5; }
endclass
```

Will this randomize? Justify your answer.

81. **Write a constraint using implication (-¿) where variable ‘mode‘ decides the range of another variable ‘data‘.**

82. **What is the role of ‘soft‘ constraints in solving conflicting conditions? Give example with conflicting hard and soft constraints.**

83. **Predict what happens:**

```
class test;
  rand int x;
  constraint c { x % 2 == 0; x % 3 == 0; }
endclass
```

What values are possible for x within default int range?

84. **What causes “No Solution” errors during randomization? Name at least 3 common reasons.**

85. **Predict behavior:**

```
class test;
  rand int a;
  constraint c1 { a == a + 1; }
endclass
```

Will this randomize? Why or why not?

86. **Write a constraint that creates an impossible-to-solve situation for the solver with 3 variables. Then fix it using soft constraints.**

87. **Explain the difference between pre-solve and post-solve values in SystemVerilog. How can you use them to debug solver issues?**

88. **Output Prediction: Array Constraint**

```
class test;
  rand bit [3:0] arr[4];
  constraint c1 {
    foreach(arr[i]) {
      arr[i] == i;
    }
  }
endclass
```

What are the possible outputs of ‘arr‘ and is it guaranteed to randomize?

89. **What does the constraint solver do when multiple valid solutions exist? How does it choose one? Can we control that?**

90. **Create a case where a constraint randomizes in 90% of cases and fails in 10%. Why might this be useful?**

91. **Predict output: Conditional Constraint**

```
class test;
  rand bit [3:0] a, b;
  constraint c1 { if (a > 5) b < 3; }
endclass
```

Will this work without error? Why? What if ‘a ¡= 5‘?

92. **Design a constraint where the sum of 5 variables must be 100, but each variable is limited between 0 to 15. Will it succeed? Why?**

93. **What's the output of this constraint with uniqueness?**

```
class test;
  rand bit [3:0] a, b, c;
  constraint c1 { unique {a, b, c}; }
endclass
```

What happens if ‘a = 4‘, ‘b = 4‘, ‘c = 4‘ was pre-assigned before randomization?

94. **What are the effects of ‘solve before‘ in constraint solving order? Show an example where improper solving order leads to unsolvable constraints.**

95. **Predict output: Using ‘inside‘ with invalid range**

```
class test;
  rand int x;
  constraint c1 { x inside { [5:1] }; }
endclass
```

Does this work? Why or why not? What fixes it?

96. **Explain constraint inheritance and how it could cause constraint duplication or unintended overrides. Show example.**

97. **Is the following valid? What will it do?**

```
constraint c1 { 1 inside { [1:3] }; }
```

98. **What is the solver strategy when two equally valid 'soft' constraints contradict each other? Which one wins?**

99. **Write a test case where 'randc' behaves differently than 'rand'. Why is this behavior important in test generation?**

100. **Predict result:**

```
class test;
  rand bit [3:0] a;
  constraint c1 { a < 10; a > 20; }
endclass
```

What will be the result of 'randomize()'? Explain in terms of solver logic.

Section 5: OOP + Polymorphism + Factory Concepts (Q101–125)

101. **What is polymorphism in SystemVerilog? How is it different from method overloading?**

102. **What are virtual methods? Why are they necessary in a verification environment?**

103. **Give an example of dynamic polymorphism using classes and virtual methods.**

104. **What is the difference between virtual class and abstract class in SystemVerilog?**

105. **Can we create an object of a virtual class? Why or why not?**

106. **What is the role of the 'super' keyword in inheritance? Show with an example.**

107. **What happens if a child class redefines a virtual method but forgets to use the ‘virtual‘ keyword?**

108. **Predict output: Virtual Method Call**

```
class A;
  virtual function void display();
    $display("A");
  endfunction
endclass

class B extends A;
  function void display();
    $display("B");
  endfunction
endclass

A a = new B();
a.display();
```

109. **Explain dynamic casting with $cast and is_a(). Provide examples.**

110. **What is method overriding? How does it differ from overloading?**

111. **Can we have multiple classes with the same method name but different arguments in SV? What is the behavior?**

112. **What is the syntax of creating a parameterized class? Show an example.**

113. **Can constructors be inherited in SystemVerilog? Explain how constructors are chained.**

114. **How do you override a base class method in a child class? What happens if ‘virtual‘ is missing?**

115. **What is the difference between ‘virtual‘ and ‘pure virtual‘ functions?**

116. **Explain the factory concept in SystemVerilog. Why is it used in UVM?**

117. **Give a simple implementation of a manual factory using base and derived classes.**

118. **Write a code that shows how the factory chooses the correct implementation at runtime.**

119. **Predict output: Inheritance Chain**

```
class Base;
  virtual function void f();
    $display("Base");
  endfunction
```

```
endclass

class Mid extends Base;
  function void f();
    $display("Mid");
  endfunction
endclass

class Top extends Mid;
  function void f();
    $display("Top");
  endfunction
endclass

Base b = new Top();
b.f();
```

120. **What is deep copy and shallow copy in OOP? How can you implement deep copy in SystemVerilog?**

121. **How to clone a class object in SystemVerilog? Is it deep or shallow by default?**

122. **What is the use of ‘extern‘ functions in classes? Give pros and cons.**

123. **Write a program using ‘extern‘ to define methods outside the class body.**

124. **How does constructor overloading work in SystemVerilog?**

125. **What is ‘typedef class‘ used for? Give an example where it’s beneficial.**

Theory-Based Questions

Q126. List and explain the difference between logical and bitwise operators in SystemVerilog.

Q127. What is the purpose of reduction operators? Provide examples.

Q128. Describe the result of using ‘==‘ vs ‘===‘ in comparisons.

Q129. Explain the difference between >>, >>> and <<.

Q130. How do replication and streaming operators work? Give syntax and use cases.

Q131. What is the default data type of constants in expressions?

Q132. How does operator precedence impact the result of a mixed-type expression?

Q133. Discuss the behavior of the ternary operator in SV. Can it be nested?

Q134. Explain overflow in arithmetic operations with unsigned vs signed operands.

Q135. Can you override operator behavior in SystemVerilog?

Conceptual + Output Prediction

Q136. Predict the output:

```
logic [3:0] a = 4'b1010;
logic [3:0] b = 4'b0101;
$display("%b", a & b);
```

Q137. What is the output of:

```
int a = 5;
int b = 2;
$display("%0d", a / b);
```

Q138. Describe the result of:

```
logic signed [7:0] x = -5;
logic signed [7:0] y = x >>> 1;
```

Q139. Evaluate and explain:

```
logic [3:0] data = 4'b1100;
$display("%b", ~|data);
```

Q140. Predict the output:

```
logic [3:0] a = 4'b0001;
logic [3:0] b = {4{a}};
```

Q141. What happens when:

```
logic [3:0] a = 4'bz001;
logic b = a === 4'bz001;
```

Q142. Difference in behavior:

```
logic a = 1'bx;
$display("%b", a == 1'bx); // vs ===
```

Q143. Predict:

```
logic [3:0] x = 4'b0110;
logic y = &x;
```

Q144. Output of this conditional:

```
int a = 3, b = 5;
int max = (a > b) ? a : b;
```

Q145. Operator precedence quiz:

```
int x = 3 + 5 * 2;
```

Practice Coding Challenge

Q146. Write a program to use replication operator to construct an 8-bit pattern from 2-bit input.

Q147. Write a function to calculate parity using reduction XOR ().

Q148. Use logical operators to simulate a basic AND gate with error catching.

Q149. Simulate a priority encoder using shift and mask operators.

Q150. Create a 32-bit barrel shifter using concatenation and dynamic shifting.

Focus: Randomization & 'inside' Operator

SystemVerilog provides randomization capabilities that allow generating data with constraints and conditions. The 'inside' operator allows checking if a value exists within a set or range.

Theory and Conceptual Questions

Q151. What is the purpose of the 'rand' and 'randc' keywords in SystemVerilog?

Q152. Differentiate between 'rand' and 'randc' variables with examples.

Q153. Explain the role of 'constraint' blocks in randomization.

Q154. How does the 'inside' operator work? Provide syntax and use case.

Q155. Compare the 'inside' operator with 'case' and 'if' statements.

Q156. Can we use dynamic arrays and queues in constraints?

Q157. What is the behavior of 'randc' when the full cycle is not consumed?

Q158. Can a variable be both 'rand' and 'static'?

Q159. Explain 'solve before' in constraint ordering.

Q160. List the methods to call randomization inside a class.

Output-Based and Edge Case Scenarios

Q161. Predict output:

```
class A;
  rand bit [3:0] val;
  constraint range_c { val inside {[4:9]}; }
endclass

A obj = new();
obj.randomize();
$display("val = %0d", obj.val);
```

Q162. What happens if:

```
  constraint empty { val inside {}; }
```

Q163. Given:

```
  randc int a;
  constraint val_c { a inside {1, 3, 5}; }
```

How many unique randomizations before repetition?

Q164. What's the issue here?

```
  rand int val;
  constraint c { val inside {[5:10], [15:12]}; }
```

Q165. What's the output?

```
class InsideTest;
  rand int num;
  constraint c { num inside {2, 4, 6, 8}; }
endclass

InsideTest t = new();
repeat(4) begin
  t.randomize();
  $display("num = %0d", t.num);
end
```

Practical Coding and Challenges

Q166. Write a class that generates even numbers between 10 and 100 using ‘inside‘ operator.

Q167. Create a constraint to generate a value not inside ‘3, 5, 7‘.

Q168. Write a constraint to randomize a ‘randc‘ variable to values ‘1, 2, 4‘ only.

Q169. Build a test that uses ‘inside‘ to select only prime numbers from 1–30.

Q170. Use ‘randc‘ to randomize unique digits of a 4-digit number (e.g., 1234, 4052).

Q171. Design a constraint using ‘solve before‘ to ensure ‘width ¿ height‘ when both are randomized.

Q172. Write a testbench that fails if a random number does not belong to a valid list (use ‘inside‘).

Q173. Write a reusable ‘function bit check_inside(int val);‘ using ‘inside‘.

Q174. Create a class where a random color is chosen only from a user-defined set.

Q175. Simulate a dice roll using randomization. Display frequency of each face after 1000 rolls.

Focus: Dynamic Arrays, Queues, Associative Arrays

SystemVerilog provides multiple collection types like dynamic arrays, queues, and associative arrays with versatile access and storage capabilities. Each has distinct use cases and performance characteristics.

Theory and Conceptual Questions

Q176. Define dynamic arrays and how they differ from static arrays.

Q177. What is the default size of a dynamic array after declaration?

Q178. What are the key methods associated with queues in SystemVerilog?

Q179. Compare dynamic arrays and queues.

Q180. Explain associative arrays. What types of indices can be used?

Q181. Can associative arrays be indexed with string or enum? Provide examples.

Q182. Explain ‘*size*()‘, ‘display()‘, and ‘$exists()‘ in context of associative arrays.

Q183. Which data structure is best suited for sparse memory-like behavior?

Q184. Explain resizing of dynamic arrays with ‘new[]‘.

Q185. What are the consequences of using an uninitialized dynamic array?

Output Prediction and Trick Questions

Q186. Predict output:

```
int da[] = new[5];
foreach (da[i]) da[i] = i * 2;
$display("%p", da);
```

Q187. Given:

```
int q[$];
q = {1, 2, 3};
q.push_front(0);
q.pop_back();
$display("%p", q);
```

Q188. What is the output?

```
int aa[string];
aa["x"] = 99;
$display("val: %0d", aa["y"]);
```

Q189. What is returned by:

```
int aa[int];
$display("%0d", aa.num());
```

Q190. How many elements will this loop execute?

```
int aa[string] = '{"a":1, "b":2};
foreach(aa[i])
  $display(i, aa[i]);
```

Practical Coding Exercises

Q191. Write a class that initializes a dynamic array of size 10 with values equal to square of index.

Q192. Implement a queue that stores the last 5 inputs (rolling buffer).

Q193. Write a function that returns the maximum element in a queue.

Q194. Create a class that uses an associative array to store and retrieve employee names by ID.

Q195. Write a task that prints key-value pairs of an associative array with string index.

Q196. Create a scoreboard using associative array with testcase name as key and pass/fail as value.

Q197. Build a sorting function for a queue of integers (any basic sorting algorithm).

Q198. Create a method to clear all entries of an associative array.

Q199. Simulate a real-time task queue using ‘push_back‘ and ‘pop_front‘ methods.

Q200. Write a class method to merge two dynamic arrays into a new array and remove duplicates.

Q201. Explain the concept of a thread in SystemVerilog.

Q202. How does ‘fork-join‘ work in SystemVerilog? What is its role in multi-threaded execution?

Q203. Compare fork-join with ‘fork-join_any‘ and ‘fork-join_none‘.

Q204. What is the difference between ‘join‘ and ‘join_any‘? When would you use one over the other?

Q205. How does ‘join_none‘ work in the context of fork-join? Provide an example.

Q206. What is the purpose of ‘wait‘ and ‘wait fork‘ in SystemVerilog?

Q207. What is an event in SystemVerilog? How does it synchronize threads?

Q208. Describe the ‘set‘, ‘wait‘, and ‘clear‘ methods used with events.

Q209. What happens if a thread is ‘joined‘ after the ‘join‘ statement is called?

Q210. How does a semaphore work in SystemVerilog? What is its role in synchronization?

Q211. Can you use semaphores in SystemVerilog for mutual exclusion? How?

Q212. Explain the differences between binary semaphores and counting semaphores.

Q213. How would you use semaphores to protect shared resources?

Q214. What is a race condition in the context of threads and how can it be avoided?

Q215. When would you use ‘event‘ vs ‘semaphore‘ in testbenches?

Q216. Explain the purpose of ‘fork‘ and how multiple threads can be launched using it.

Q217. What is the potential risk of using ‘fork-join‘ in a simulation?

Q218. Discuss the importance of thread synchronization in UVM (Universal Verification Methodology).

Q219. How do you handle deadlock scenarios when using semaphores?

Q220. Describe a real-world scenario where you would use ‘wait‘ and ‘join‘.

Output Prediction and Trick Questions

Q221. What will be the output for the following code?

```
initial begin
  fork
    $display("Thread 1");
    $display("Thread 2");
  join
end
```

Q222. Predict output for:

```
int count = 0;
event e;
initial begin
  fork
    count = count + 1;
    wait (e);
    count = count + 2;
  join
  $display("count = %0d", count);
end
```

Q223. What is the behavior if:

```
event start;
initial begin
  fork
    wait(start);
    $display("Event triggered in thread 1");
  join_any
  $display("Thread finished");
end
start = 1;
```

Q224. What will the following code display?

```
event start;
initial begin
  fork
    wait(start);
    $display("Thread A");
  join_none
  $display("Thread B");
end
start = 1;
```

Q225. What happens when you use ‘wait‘ inside a fork-join block but the condition never becomes true?

```
initial begin
  fork
    wait (count > 10);
    $display("Condition met");
  join
end
```

Practical Coding Exercises

Q226. Write a code that uses a semaphore to protect a shared resource (e.g., a counter) between multiple threads.

Q227. Design a task using fork-join to perform a time-consuming operation (e.g., download data from multiple sources).

Q228. Use an event to synchronize the completion of multiple threads before starting another operation.

Q229. Create a deadlock situation using semaphores and explain how to resolve it.

Q230. Write a task where multiple threads are synchronized using ‘wait‘ and ‘join_any‘ based on an event.

Q231. Implement a function where ‘fork‘ creates three threads performing different operations and then all threads synchronize.

Q232. Use ‘join_none‘ to run multiple threads concurrently without waiting for them to finish.

Q233. Create a code that demonstrates the use of ‘wait‘ with ‘join‘ to ensure that a specific thread finishes before others.

Q234. Write an example using multiple threads, events, and semaphores to model a producer-consumer scenario.

Q235. Implement a sequence generator using ‘fork-join‘ where each thread generates a part of the sequence concurrently.

Q236. What will be the output for the following code?

```
initial begin
  fork
    $display("Thread 1");
    #5 $display("Thread 2");
  join
  $display("End");
end
```

Q237. Predict the output for the code below:

```
initial begin
  fork
    $display("First");
    #10 $display("Second");
    $display("Third");
  join
end
```

Q238. What is the output when using a ‘join_any‘?

```
initial begin
  fork
    $display("Thread A");
    #5 $display("Thread B");
  join_any
  $display("End");
end
```

Q239. Predict the output for the following code that uses ‘fork-join_none‘:

```
initial begin
  fork
    #5 $display("First");
    #10 $display("Second");
  join_none
  $display("End");
end
```

Q240. What happens when multiple threads are started with a ‘join _any‘ and only one of the threads completes? Predict the output:

```
initial begin
  fork
    #5 $display("Thread 1");
    #10 $display("Thread 2");
    #15 $display("Thread 3");
  join_any
  $display("End");
end
```

Q241. What is the result of the following ‘fork-join‘ structure, especially focusing on the thread completion timing?

```
initial begin
  fork
    #5 $display("Thread 1");
    #15 $display("Thread 2");
    #10 $display("Thread 3");
  join
  $display("Finished");
end
```

Q242. Predict the output of the following code, with a focus on the order of execution:

```
initial begin
  fork
    $display("First");
    #20 $display("Second");
    $display("Third");
  join
  $display("Last");
end
```

Q243. Predict the output for the following scenario using ‘fork-join_none‘:

```
initial begin
  fork
    $display("Thread 1");
    #5 $display("Thread 2");
    #10 $display("Thread 3");
  join_none
  $display("Thread End");
end
```

Q244. What happens if you use a ‘join_any‘ with two threads, but only one thread waits indefinitely?

```
initial begin
  fork
    #10 $display("Thread 1");
    wait(1);
  join_any
  $display("End");
end
```

Q245. How would you predict the output for the following code using ‘fork-join_none‘?

```
initial begin
  fork
    #5 $display("A");
```

```
    #10 $display("B");
  join_none
  $display("C");
  $display("D");
end
```

Q246. What happens when you use a ‘join‘ for one thread while another thread is delayed indefinitely? Predict the output:

```
initial begin
  fork
    #5 $display("Thread 1");
    #100 $display("Thread 2");
  join
  $display("All threads completed");
end
```

Q247. Predict the output when ‘join_any‘ is used with multiple threads where one thread takes significantly longer than others:

```
initial begin
  fork
    #20 $display("Thread 1");
    #10 $display("Thread 2");
    #50 $display("Thread 3");
  join_any
  $display("End");
end
```

Q248. How would the following code behave with ‘join‘? Focus on the sequence of displays:

```
initial begin
  fork
    $display("Thread 1");
    #5 $display("Thread 2");
  join
  $display("Finished");
end
```

Q249. What would be the output of the following code that involves ‘join_any‘?

```
initial begin
  fork
    #10 $display("First");
    #5 $display("Second");
  join_any
  $display("Third");
end
```

Q250. Predict the output when ‘join_none‘ is used with multiple threads:

```
initial begin
  fork
    #3 $display("A");
    #6 $display("B");
  join_none
  $display("C");
end
```

Q251. What will be the output for the following code that uses randomization with a simple constraint?

```
class RandomExample;
  rand bit [3:0] x;
  constraint c1 {x > 5;}
endclass

initial begin
  RandomExample obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q252. How would you predict the output of the following code that involves multiple random variables with dependencies?

```
class RandomDependent;
  rand bit [3:0] x, y;
  constraint c1 {x < y;}
endclass

initial begin
  RandomDependent obj = new();
  obj.randomize();
  $display("x = %0d, y = %0d", obj.x, obj.y);
end
```

Q253. What happens when a random variable is constrained within a range using ‘randomize‘?

```
class RandomRange;
  rand bit [7:0] data;
  constraint c1 {data inside {8'd10, 8'd20, 8'd30};}
endclass

initial begin
```

```
  RandomRange obj = new();
  obj.randomize();
  $display("data = %0d", obj.data);
end
```

Q254. What will be the result of the randomization with a complex constraint that uses an ‘if‘ statement?

```
class RandomComplex;
  rand bit [3:0] a, b;
  constraint c1 {a < 5; if (a == 2) b > 3;}
endclass

initial begin
  RandomComplex obj = new();
  obj.randomize();
  $display("a = %0d, b = %0d", obj.a, obj.b);
end
```

Q255. In the following code, what will be the output when randomization is constrained with an ‘else‘ block?

```
class RandomElse;
  rand bit [3:0] x, y;
  constraint c1 {x < 5; else y > 5;}
endclass

initial begin
  RandomElse obj = new();
  obj.randomize();
  $display("x = %0d, y = %0d", obj.x, obj.y);
end
```

Q256. What happens when randomization fails with the following code and how would you debug it?

```
class RandomFailure;
  rand bit [3:0] p;
  constraint c1 {p > 5;}
endclass

initial begin
  RandomFailure obj = new();
  if (!obj.randomize()) $display("Randomization failed");
  else $display("p = %0d", obj.p);
end
```

Q257. What will be the output when randomization is attempted on multiple variables with conflicting constraints?

```
class RandomConflict;
  rand bit [3:0] a, b;
  constraint c1 {a > b;}
  constraint c2 {b > 2;}
endclass

initial begin
  RandomConflict obj = new();
  if (!obj.randomize()) $display("Randomization failed");
  else $display("a = %0d, b = %0d", obj.a, obj.b);
end
```

Q258. What is the expected output when multiple randomization constraints are applied with ‘unique‘?

```
class RandomUnique;
  rand bit [3:0] a, b, c;
  constraint c1 {a != b; a != c;}
endclass

initial begin
  RandomUnique obj = new();
  obj.randomize();
  $display("a = %0d, b = %0d, c = %0d", obj.a, obj.b, obj.c);
end
```

Q259. How would you predict the result of randomization involving a ‘dist‘ constraint?

```
class RandomDist;
  rand bit [3:0] x;
  constraint c1 {x dist {1:2, 5:3, 10:1};}
endclass

initial begin
  RandomDist obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q260. What will happen if you use a ‘constraint‘ with ‘inside‘ for a dynamic array?

```
class RandomDynamic;
  rand bit [7:0] data [10:0];
  constraint c1 {data inside {[10:20, 30:40]};}
```

```
endclass

initial begin
  RandomDynamic obj = new();
  obj.randomize();
  $display("data = %0p", obj.data);
end
```

Q261. Predict the result of the following randomization with an 'if' condition in the constraint.

```
class RandomIfCondition;
  rand bit [7:0] x, y;
  constraint c1 {if (x > 10) y < 5;}
endclass

initial begin
  RandomIfCondition obj = new();
  obj.randomize();
  $display("x = %0d, y = %0d", obj.x, obj.y);
end
```

Q262. What is the expected output when a class with multiple variables and constraints is randomized?

```
class RandomMultiVar;
  rand bit [7:0] x, y, z;
  constraint c1 {x < 10; y > 20; z inside {10, 20, 30};}
endclass

initial begin
  RandomMultiVar obj = new();
  obj.randomize();
  $display("x = %0d, y = %0d, z = %0d", obj.x, obj.y, obj.z);
end
```

Q263. How does the 'randomize()' method behave when constraints conflict? What is the expected output in the following case?

```
class RandomConflict;
  rand bit [7:0] x;
  constraint c1 {x > 5;}
  constraint c2 {x < 10;}
endclass

initial begin
  RandomConflict obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q264. What will be the outcome if randomization fails due to impossible constraints?

```
class RandomImpossible;
  rand bit [7:0] x;
  constraint c1 {x > 100;}
  constraint c2 {x < 50;}
endclass

initial begin
  RandomImpossible obj = new();
  if (!obj.randomize()) $display("Randomization failed");
  else $display("x = %0d", obj.x);
end
```

Q265. How can you use a ‘with‘ constraint to limit the randomization range in a class?

```
class RandomWith;
  rand bit [7:0] x;
  constraint c1 {x with {x > 5 && x < 20};}
endclass

initial begin
  RandomWith obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q266. What will happen when multiple constraints use a ‘forever‘ block in the randomization?

```
class RandomForever;
  rand bit [7:0] x;
  constraint c1 {forever {x > 5;}}
endclass

initial begin
  RandomForever obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q267. Predict the result of the following code with randomization and the ‘recurse‘ constraint.

```
class RandomRecurse;
  rand bit [3:0] x;
  constraint c1 {recurse {x != 2;}};
```

```
endclass

initial begin
  RandomRecurse obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q268. How does a ‘constraint‘ with ‘select‘ behave during randomization?

```
class RandomSelect;
  rand bit [7:0] x, y;
  constraint c1 {select x > y;}
endclass

initial begin
  RandomSelect obj = new();
  obj.randomize();
  $display("x = %0d, y = %0d", obj.x, obj.y);
end
```

Q269. How does the ‘constraint‘ with ‘foreach‘ work for randomizing dynamic arrays?

```
class RandomForeach;
  rand bit [7:0] data [10:0];
  constraint c1 {foreach (data[i]) data[i] < 50;}
endclass

initial begin
  RandomForeach obj = new();
  obj.randomize();
  $display("data = %0p", obj.data);
end
```

Q270. What happens if you try to randomize a variable that is not properly constrained?

```
class RandomNoConstraint;
  rand bit [3:0] x;
endclass

initial begin
  RandomNoConstraint obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q271. What is the behavior when constraints in a class use random variables with ‘==‘ and ‘!=‘ operators?

```
class RandomEquality;
  rand bit [3:0] a, b;
  constraint c1 {a != b;}
endclass

initial begin
  RandomEquality obj = new();
  obj.randomize();
  $display("a = %0d, b = %0d", obj.a, obj.b);
end
```

Q272. How does randomization with '$dist_uniform' work? Predict the result for the following code:

```
class RandomDistUniform;
  rand bit [7:0] x;
  constraint c1 {x dist {1:10, 5:30, 10:10};}
endclass

initial begin
  RandomDistUniform obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q273. What will be the result of randomization when 'inside' is used for a bit vector in a constraint?

```
class RandomInside;
  rand bit [7:0] data;
  constraint c1 {data inside {[10:20, 30:40]};}
endclass

initial begin
  RandomInside obj = new();
  obj.randomize();
  $display("data = %0d", obj.data);
end
```

Q274. How do you handle constraints with 'randc' (randomization with coverage)? Predict the output.

```
class RandomRandc;
  randc bit [7:0] x;
  constraint c1 {x > 10;}
endclass
```

```
initial begin
  RandomRandc obj = new();
  obj.randomize();
  $display("x = %0d", obj.x);
end
```

Q275. What is the expected behavior of randomization with complex constraints using ‘unique‘ and ‘priority‘?

```
class RandomPriority;
  rand bit [7:0] a, b;
  constraint c1 {unique a > b;}
  constraint c2 {priority b > 10;}
endclass

initial begin
  RandomPriority obj = new();
  obj.randomize();
  $display("a = %0d, b = %0d", obj.a, obj.b);
end
```

Q276. What is the difference between static and dynamic arrays in SystemVerilog?

Q277. How would you declare a dynamic array and assign values to it in SystemVerilog?

Q278. Explain the concept of associative arrays and their use in SystemVerilog.

Q279. How do you initialize an array in SystemVerilog with all elements set to zero?

Q280. How can you resize a dynamic array at runtime in SystemVerilog?

Q281. What will happen if you try to assign an associative array without initializing the index?

Q282. How do you iterate through the elements of a dynamic array in SystemVerilog?

Q283. What will be the output of the following code snippet in SystemVerilog?

```
int arr[5] = '{1, 2, 3, 4, 5};
$display("%d", arr[2]);
```

Q284. How would you find the number of elements in a dynamic array?

Q285. Explain how the ‘new‘ operator is used in dynamic arrays.

Q286. What is the result of using the ‘size‘ method on a dynamic array?

Q287. How do you initialize a multidimensional dynamic array in SystemVerilog?

Q288. What will be the output of this SystemVerilog code involving multidimensional arrays?

```
int arr[2:0][3:0] = '{'{1, 2, 3, 4}, '{5, 6, 7, 8}, '{9, 10, 11, 12}};
$display("%d", arr[1][2]);
```

Q289. How do you declare an associative array in SystemVerilog and assign values using string keys?

```
string arr[string];
arr["key1"] = "value1";
arr["key2"] = "value2";
```

Q290. Can you use an array as an argument to a function in SystemVerilog? If so, how?

Q291. What is the difference between an array and a queue in SystemVerilog?

Q292. How do you concatenate two arrays in SystemVerilog?

Q293. What is the result of using array1 = array2; for dynamic arrays in SystemVerilog?

Q294. How would you delete a dynamic array in SystemVerilog?

Q295. Can you access a specific element in an associative array using a non-existent key?

Q296. How do you check if an associative array contains a specific key in SystemVerilog?

Q297. What is the output of this code, which accesses an element of a dynamic array using an out-of-bound index?

```
int arr[3] = '{1, 2, 3};
$display("%d", arr[5]);
```

Q298. How do you perform a deep copy of an array in SystemVerilog?

Q299. What is the behavior of an associative array when all its elements are removed?

Q300. Can you assign one dynamic array to another dynamic array with different sizes? What happens if the arrays are of different sizes?

Q301. How do you find the largest element in a dynamic array?

Q302. What will the following code output in SystemVerilog when trying to assign an empty dynamic array to a new array?

```
int arr[] = new[5];
int new_arr[];
new_arr = arr;
$display("%d", new_arr.size());
```

Q303. How would you use an associative array in a for-each loop?

```
foreach (arr[key]) begin
  $display("key = %s, value = %s", key, arr[key]);
end
```

Q304. Explain the concept of ‘randc‘ with arrays in SystemVerilog.

Q305. Can a dynamic array contain a variable of a class type in SystemVerilog?

Q306. How can you initialize a packed array in SystemVerilog?

Q307. Explain how to use a queue in SystemVerilog and the differences with arrays.

Q308. What is the effect of using ‘+=‘ in a queue assignment in SystemVerilog?

Q309. How do you remove an element from the front of a queue in SystemVerilog?

Q310. Can you access elements of a queue directly by index in SystemVerilog? If so, how?

Q311. How would you perform a binary search on a dynamic array in SystemVerilog?

Q312. What will be the output of the following code involving multidimensional arrays?

```
int arr[2][3] = '{'{1, 2, 3}, '{4, 5, 6}};
$display("%d", arr[1][0]);
```

Q313. How do you find the sum of all elements in an array in SystemVerilog?

Q314. How do you access the first and last elements of a queue in SystemVerilog?

Q315. Can you resize a queue dynamically? How?

Q316. What happens if you try to assign a queue to an array with a fixed size in SystemVerilog?

Q317. How would you concatenate two queues in SystemVerilog?

Q318. Can you use an array of queues in SystemVerilog? If so, how would you declare it?

Q319. How do you clear a queue in SystemVerilog?

Q320. Can you compare two arrays for equality in SystemVerilog? If so, how?

Q321. How can you sort an array in SystemVerilog?

Q322. What is the result of using the ‘find()‘ method on an associative array in SystemVerilog?

Q323. How do you use ‘foreach‘ for array of queues in SystemVerilog?

Q324. How do you get the last index of a dynamic array in SystemVerilog?

Q325. Can you use a dynamic array inside a ‘foreach‘ loop in SystemVerilog? If yes, how?

Q326. Can a class be instantiated without using the ‘new‘ keyword in SystemVerilog? Explain with an example.

Q327. What is the difference between a ‘static‘ and a ‘dynamic‘ class property in SystemVerilog?

Q328. What will happen if you try to assign a class object to another object without using the ‘new‘ keyword in SystemVerilog?

Q329. How does inheritance work in SystemVerilog classes, and what happens if you don’t define a constructor in the base class?

Q330. How does polymorphism work in SystemVerilog? Provide an example where polymorphism might cause unexpected behavior.

Q331. What is the effect of calling a base class method from a derived class object in SystemVerilog? Explain with code.

Q332. Can you call a method of the derived class from the base class object in SystemVerilog? Justify your answer.

Q333. Can a virtual function in SystemVerilog be called without an object instance? What happens if you do so?

Q334. What happens if a virtual function is called on a base class object, but the object is actually a derived class?

Q335. Can you declare a class member with the ‘local‘ keyword? What does this imply for object visibility?

Q336. What will happen if you try to create an object of a class that contains a pure virtual function in SystemVerilog?

Q337. How do you resolve ambiguity when multiple classes in an inheritance hierarchy contain a method with the same name?

Q338. What is the significance of using ‘this‘ keyword in a class method in SystemVerilog?

Q339. Can a class method be static and virtual at the same time in SystemVerilog? Explain why or why not.

Q340. What happens if you try to instantiate an object from an abstract class in SystemVerilog?

Q341. Can a base class object access derived class members? Justify with an example.

Q342. What will happen if you try to assign a base class object to a derived class object reference?

Q343. What is the difference between deep and shallow copy for class objects in SystemVerilog?

Q344. How do you handle the destruction of a class object? What happens if a class does not define a destructor?

Q345. Can you use ‘super‘ to access a derived class method from the base class? Explain the behavior in SystemVerilog.

Q346. How does memory management work for class objects in SystemVerilog? Is garbage collection automatic or manual?

Q347. Can you have multiple constructors in SystemVerilog classes? What is the effect of defining multiple constructors?

Q348. How can you handle a situation where a class inherits from multiple classes with conflicting member names?

Q349. What happens when you override a virtual method in SystemVerilog and call it through a pointer to the base class?

Q350. Is it possible to create an object of a derived class without calling the constructor of the base class? Explain.

Q351. How do ‘new‘ and ‘super.new‘ differ when used in constructors of derived classes in SystemVerilog?

Q352. What will happen if you call a method that has been overridden but the object is of the base class type?

Q353. What are the benefits and drawbacks of using ‘virtual‘ and ‘pure virtual‘ functions in SystemVerilog?

Q354. Can you use a base class pointer to refer to a derived class object? Explain the implications for method calls.

Q355. What happens if you declare a ‘static‘ member in a SystemVerilog class? How is it accessed?

Q356. Can a class method be static and virtual? How would this work in SystemVerilog?

Q357. How do you prevent a class method from being overridden in SystemVerilog? Provide an example.

Q358. What happens when a virtual function is overridden in SystemVerilog but it is not called via the base class reference?

Q359. Can you pass a class object as a function argument? If yes, explain the method of passing and returning class objects in SystemVerilog.

Q360. What will happen if you instantiate a class inside a function in SystemVerilog?

Q361. What is the purpose of using ‘virtual‘ keyword for a function inside a class, and how does it affect the class hierarchy?

Q362. What is the result when you use ‘new‘ for an abstract class in SystemVerilog?

Q363. Can a ‘static‘ function access non-static data members in a class in SystemVerilog?

Q364. How would you handle multiple constructors in a class, including default constructors and parameterized constructors?

Q365. How would the following code behave in SystemVerilog?

```
class Base;
    virtual function void show();
        $display("Base show");
    endfunction
endclass

class Derived extends Base;
    virtual function void show();
        $display("Derived show");
    endfunction
endclass

Base b = new;
Derived d = new;
b = d;
b.show();
```

Q366. How would you use ‘super‘ in SystemVerilog to call a method of the base class from the derived class?

Q367. What would happen if a base class pointer refers to a derived class object, but the method is not virtual in SystemVerilog?

Q368. What is the outcome of calling a function with a class object that is not initialized?

Q369. How can you initialize class data members within the constructor in SystemVerilog?

Q370. Explain the concept of method overriding with an example. What happens if the method is not marked as ‘virtual‘ in SystemVerilog?

Q371. Can you use a class method without instantiating a class object in SystemVerilog? If so, how?

Q372. Can you instantiate an object of a base class if the base class is abstract and has a virtual method? Justify your answer.

Q373. Can you use a ‘foreach‘ loop to iterate over class members? Provide an example.

Q374. How can you implement multiple inheritance in SystemVerilog classes? Discuss the complications involved.

Q375. How would the following code behave if ‘@‘ is used in a class method call in SystemVerilog?

```
class A;
    function void func();
        $display("Class A");
    endfunction
endclass
```

Q376. Can you assign a class type variable to a variable of the base class type? What will happen?

Q377. What is the purpose of ‘virtual‘ and ‘pure virtual‘ in SystemVerilog classes, and how do they interact with inheritance?

Q378. What will happen if you call an unimplemented method in a derived class that is marked as pure virtual in SystemVerilog?

Q379. Can a class constructor take arguments in SystemVerilog? If yes, explain with an example.

Q380. What are the implications of using a ‘pure virtual function‘ in an interface versus a class in SystemVerilog?

Q381. How would the output change if you modify a derived class constructor and do not call ‘super.new‘ in SystemVerilog?

Q382. How would you define an interface and a class that uses the interface methods?

Q383. How do ‘initial‘ and ‘always‘ blocks affect class objects in SystemVerilog simulations?

Q384. Can you use an interface with a class object in SystemVerilog? Provide an example.

Q385. What will happen if you use ‘super‘ with a static method inside a class in SystemVerilog?

Q386. What is the significance of the ‘virtual‘ keyword when used for a class method? How does it differ from regular methods?

Q387. How do you force a derived class to implement a virtual function from a base class?

Q388. How would a derived class call the base class constructor in SystemVerilog?

Q389. How can a class be used to store values in an associative array in SystemVerilog?

Q390. Can a class inherit from a class of a different type, such as a randomize-able class? What considerations should be made?

Q391. How do you simulate class objects that need to be initialized in different threads?

Q392. How can a base class method access data members from its derived classes?

Q393. Can a class contain both ‘static‘ and ‘virtual‘ methods? If so, what happens when a class inherits such methods?

Q394. What is the significance of defining a constructor and destructor in SystemVerilog classes?

Q395. Can SystemVerilog classes implement templates or generics? If so, how would you declare them?

Q396. How would you create a singleton pattern in SystemVerilog using a class?

Q397. Can ‘super‘ be used in a ‘function‘ as well as a ‘method‘ in SystemVerilog? Justify your answer.

Q398. How do you perform memory management for class objects in a SystemVerilog testbench?

Q399. What happens if you define a pure virtual function in SystemVerilog but forget to implement it in the derived class?

Q400. Can a derived class override a static method of its base class in SystemVerilog? Explain the behavior.

Q401. What will be the output of the following code snippet?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Class B display*

Q402. What will be the output if the ‘display‘ method is marked ‘virtual‘ in class A, and the method call is made through an object of class A?

```
class A;
    virtual function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass
```

```
A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Class B display*

Q403. Consider the following code. What will be the output?

```
class A;
    virtual function void show();
        $display("A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B show");
    endfunction
endclass

A a;
a = new B;
a.show();
```

Answer: *B show*

Q404. Predict the output of this code:

```
class A;
    function void display();
        $display("Class A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B");
    endfunction
endclass

A a = new;
B b = new;
a = b;
b.display();
```

Answer: *Class B*

Q405. What will the output be for the following code?

```
class A;
    virtual function void display();
        $display("A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("B display");
    endfunction
endclass

A a;
B b;
a = new B;
b = a;
b.display();
```

Answer: *B display*

Q406. What will the output be when the constructor is not called explicitly for a class object in SystemVerilog?

```
class A;
    function new();
        $display("Constructor of A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor of B");
    endfunction
endclass

A a;
a = new B;
```

Answer: *Constructor of B*

Q407. What will be the result of the following code?

```
class A;
    virtual function void show();
        $display("A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B show");
    endfunction
endclass

A a;
B b;
a = new B;
b = new A;
a.show();
```

Answer: *B show*

Q408. What will be the output of the following code?

```
class A;
    function void display();
        $display("Display A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Display B");
    endfunction
endclass

A a = new;
a = new B;
a.display();
```

Answer: *Display B*

Q409. Consider this code. What will the output be?

```
class A;
    function void show();
        $display("A show");
    endfunction
```

```
endclass

class B extends A;
    function void show();
        $display("B show");
    endfunction
endclass

A a;
a = new B;
a.show();
```

Answer: *B show*

Q410. Predict the output when 'super' is used in the derived class method:

```
class A;
    function void display();
        $display("Display A");
    endfunction
endclass

class B extends A;
    function void display();
        super.display();
        $display("Display B");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Display A, Display B*

Q411. What is the output of the following code when the constructor of class 'B' is called?

```
class A;
    function new();
        $display("Constructor of A");
    endfunction
endclass

class B extends A;
    function new();
```

```
        $display("Constructor of B");
    endfunction
endclass

A a = new B;
```

Answer: *Constructor of B*

Q412. Predict the result of the following code:

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

B b = new;
A a = b;
a.display();
```

Answer: *Class B display*

Q413. What will be the output of the following code?

```
class A;
    virtual function void show();
        $display("Class A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("Class B show");
    endfunction
endclass

A a;
a = new B;
a.show();
```

Answer: *Class B show*

Q414. Consider this code. What will be the output?

```
class A;
    function void display();
        $display("A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("B display");
    endfunction
endclass

A a;
B b;
a = b;
b.display();
```

Answer: *B display*

Q415. What will the output be when you call a base class method from a derived class object using 'super'?

```
class A;
    function void display();
        $display("Display A");
    endfunction
endclass

class B extends A;
    function void display();
        super.display();
        $display("Display B");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Display A, Display B*

Q416. What happens if you try to instantiate an object of a class that has an abstract method?

```
class A;
    virtual function void display();
        $display("Display A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Display B");
    endfunction
endclass

A a = new B;
```

Answer: *Display B*

Q417. What will be the output for the following code?

```
class A;
    function void display();
        $display("A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("B display");
    endfunction
endclass

A a;
B b;
a = b;
a.display();
```

Answer: *B display*

Q418. What is the result of the following code?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
```

```
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a;
a = new B;
a.display();
```

Answer: *Class B display*

Q419. What will be printed when the following code is executed?

```
class A;
    function void display();
        $display("Class A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B");
    endfunction
endclass

class C extends B;
    function void display();
        $display("Class C");
    endfunction
endclass

A a = new C;
a.display();
```

Answer: *Class C*

Q420. Consider the following code. What will the output be?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass
```

```
class B extends A;
    function void display();
        super.display();
        $display("Class B display");
    endfunction
endclass

A a = new B;
a.display();
```

Answer: *Class A display, Class B display*

Q421. What will be the output when using an abstract method in a derived class?

```
class A;
    virtual function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    virtual function void display();
        $display("Class B display");
    endfunction
endclass

A a = new B;
a.display();
```

Answer: *Class B display*

Q422. What will be the result of calling ‘super‘ to invoke a method from a base class?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        super.display();
        $display("Class B display");
    endfunction
endclass
```

```
A a = new B;
a.display();
```

Answer: *Class A display, Class B display*

Q423. Consider this scenario. What will the output be?

```
class A;
    virtual function void show();
        $display("Class A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("Class B show");
    endfunction
endclass

A a = new B;
a.show();
```

Answer: *Class B show*

Q424. What will happen when a derived class does not override a base class method?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    // No display function here
endclass

A a = new B;
a.display();
```

Answer: *Class A display*

Q425. Predict the output for the following code snippet:

```
class A;
    function void display();
        $display("Class A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B");
    endfunction
endclass

A a = new;
B b = new;
a = b;
b.display();
```

Answer: *Class B*

Q426. What is the output of the following code?

```
class A;
    function void show();
        $display("A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B show");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.show();
```

Answer: *B show*

Q427. Predict the output when a method is overridden in the derived class:

```
class A;
    virtual function void display();
        $display("Class A display");
    endfunction
```

```
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new B;
a.display();
```

Answer: *Class B display*

Q428. What will the output be when accessing a field from the base class in the derived class?

```
class A;
    int x;
    function new();
        x = 5;
    endfunction
endclass

class B extends A;
    function void display();
        $display("x = %d", x);
    endfunction
endclass

A a = new B;
a.display();
```

Answer: $x = 5$

Q429. What will happen if a method in the base class is called via an object of the derived class?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        super.display();
```

```
        $display("Class B display");
    endfunction
endclass

A a = new B;
a.display();
```

Answer: *Class A display, Class B display*

Q430. What is the output of the following code when a constructor is explicitly called?

```
class A;
    function new();
        $display("Constructor of A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor of B");
    endfunction
endclass

A a = new B;
```

Answer: *Constructor of B*

Q431. What will be the output of the following code when using an abstract class?

```
class A;
    virtual function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new B;
a.display();
```

Answer: *Class B display*

Q432. What happens if the derived class method doesn't call 'super' to invoke the base class method?

```
class A;
    function void display();
        $display("Display from A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Display from B");
    endfunction
endclass

A a = new B;
a.display();
```

Answer: *Display from B*

Q433. Consider the following code. What will be the output?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new;
a = new B;
a.display();
```

Answer: *Class B display*

Q434. What will the output be when 'virtual' is used for a function in the base class?

```
class A;
    virtual function void show();
        $display("Class A show");
    endfunction
```

```
endclass

class B extends A;
    function void show();
        $display("Class B show");
    endfunction
endclass

A a = new B;
a.show();
```

Answer: *Class B show*

Q435. Predict the output when a function from the base class is called on the derived class object:

```
class A;
    function void display();
        $display("Display A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Display B");
    endfunction
endclass

A a;
a = new B;
a.display();
```

Answer: *Display B*

Q436. What will be the output for the following code?

```
class A;
    function void display();
        $display("A's display");
    endfunction
endclass

class B extends A;
    function void display();
        super.display();
        $display("B's display");
```

```
    endfunction
endclass

A a;
a = new B;
a.display();
```

Answer: *A's display, B's display*

Q437. What will be the output of the following code?

```
class A;
    function void display();
        $display("A's display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("B's display");
    endfunction
endclass

class C extends B;
    function void display();
        $display("C's display");
    endfunction
endclass

A a = new C;
a.display();
```

Answer: *C's display*

Q438. What will happen when the following code is executed?

```
class A;
    virtual function void show();
        $display("Class A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("Class B show");
    endfunction
```

```
endclass

A a = new;
B b = new;
a = b;
a.show();
```

Answer: *Class B show*

Q439. What will be the output when a base class constructor is invoked by a derived class constructor?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

A a = new B;
```

Answer: *Constructor B*

Q440. What will be the output of the following code snippet when accessing a method from a base class?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Class B display*

Q441. Predict the output when a method is called on a base class object after assigning a derived class object:

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new A;
B b = new B;
a = b;
a.display();
```

Answer: *Class B display*

Q442. What will happen in the following scenario?

```
class A;
    virtual function void show();
        $display("A's show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B's show");
    endfunction
endclass

A a = new B;
a.show();
```

Answer: *B's show*

Q443. What is the result of this code?

```
class A;
    function void display();
        $display("Class A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B");
    endfunction
endclass

A a = new B;
a.display();
```

Answer: *Class B*

Q444. What will be printed when the following code is executed?

```
class A;
    function void show();
        $display("A's show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B's show");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.show();
```

Answer: *B's show*

Q445. What is the output when a base class constructor is called in a derived class constructor?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass
```

```
class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

A a = new B;
```

Answer: *Constructor B*

Q446. What is the output when calling a method from the derived class, which overrides a base class method?

```
class A;
    virtual function void show();
        $display("A's show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B's show");
    endfunction
endclass

A a = new B;
a.show();
```

Answer: *B's show*

Q447. What happens when an abstract method is called in a derived class?

```
class A;
    virtual function void show();
        $display("A's show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B's show");
    endfunction
endclass

A a = new B;
a.show();
```

Answer: *B's show*

Q448. What is the result when a base class method is called using a derived class object?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Class B display*

Q449. What will be the output of the following code when calling the constructor of a derived class?

```
class A;
    function new();
        $display("Constructor of A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor of B");
    endfunction
endclass

A a = new B;
```

Answer: *Constructor of B*

Q450. What will happen when the following code executes?

```
class A;
    function void display();
```

```
        $display("Class A");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Class B*

Q451. What will be the output when the constructor is called?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

B b = new;
```

Answer: *Constructor B*

Q471. What is the output when calling a constructor for a derived class object?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor B");
```

```
    endfunction
endclass

B b = new;
```

Answer: *Constructor B*

Q472. What will happen when you call a method of the base class using a derived class object?

```
class A;
    function void show();
        $display("Base class show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("Derived class show");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.show();
```

Answer: *Derived class show*

Q473. What will be the output when the following code is executed?

```
class A;
    function void display();
        $display("A's display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("B's display");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *B's display*

Q474. What will be the output when you call a method from the base class after a derived class object is assigned?

```
class A;
    virtual function void show();
        $display("Class A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("Class B show");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.show();
```

Answer: *Class B show*

Q475. What will happen if a derived class object calls a base class method?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Class B display*

Q476. What will be the output when a function from the base class is called on the derived class object?

```
class A;
    function void show();
        $display("A's show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B's show");
    endfunction
endclass

A a;
a = new B;
a.show();
```

Answer: *B's show*

Q477. What will be printed when the following code is executed?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

A a = new;
B b = new;
a = b;
```

Answer: *Constructor B*

Q478. What happens when a base class constructor is invoked from the derived class?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass
```

```
class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

A a = new B;
```

Answer: *Constructor B*

Q479. What will happen if a base class constructor is called after the derived class constructor?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

B b = new;
```

Answer: *Constructor B*

Q480. What is the output when a constructor is called for an object assigned to a base class?

```
class A;
    function new();
        $display("Constructor A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

A a = new B;
```

Answer: *Constructor B*

Q481. Predict the output when a method is called from a derived class that overrides a base class method:

```
class A;
    virtual function void show();
        $display("A's show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("B's show");
    endfunction
endclass

A a = new B;
a.show();
```

Answer: *B's show*

Q482. What happens when a derived class object is created and assigned to a base class pointer?

```
class A;
    function void display();
        $display("Class A display");
    endfunction
endclass

class B extends A;
    function void display();
        $display("Class B display");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.display();
```

Answer: *Class B display*

Q483. What will be printed when the following code runs?

```
class A;
    function new();
```

```
        $display("Constructor A");
    endfunction
endclass

class B extends A;
    function new();
        $display("Constructor B");
    endfunction
endclass

B b = new;
```

Answer: *Constructor B*

Q484. What will be the output when calling a method of a derived class, overriding a base class method?

```
class A;
    function void show();
        $display("Class A show");
    endfunction
endclass

class B extends A;
    function void show();
        $display("Class B show");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.show();
```

Answer: *Class B show*

Q485. What will be the result when an object of the derived class is assigned to a base class pointer and its method is invoked?

```
class A;
    function void show();
        $display("A's show");
    endfunction
endclass

class B extends A;
```

```
    function void show();
        $display("B's show");
    endfunction
endclass

A a = new;
B b = new;
a = b;
a.show();
```

Answer: *B's show*

Q486. What is the output of the following randomization code when it is executed?

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data > 50; data < 200; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized data: ¡Random value between 51 and 199¿*

Q487. How would you ensure that the random value of 'data' is an odd number between 10 and 100?

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data > 10 && data < 100 && data[0] == 1; }
endclass
```

Answer: *The constraint ensures that 'data' is an odd number between 11 and 99.*

Q488. What would be the output when randomizing the object 're' with the constraint 'data ¿ 0' and 'data mod 3 == 0'?

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data > 0 && data mod 3 == 0; }
```

```
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized data: Any number greater than 0 that is divisible by 3.*

Q489. What will be the randomized output for the following code considering ‘num‘ is within a certain range?

```
class RandomExample;
    rand bit [7:0] num;
    constraint num_range { num >= 100 && num <= 200; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized number: %0d", re.num);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized number: A value between 100 and 200.*

Q490. What happens when a ‘randc‘ constraint is applied to a variable?

```
class RandomExample;
    randc bit [3:0] num;
endclass

RandomExample re = new;
initial begin
    foreach (re.num) begin
        if (re.randomize())
            $display("Randomized num: %0d", re.num);
        else
            $display("Randomization failed");
    end
end
```

Answer: *'randc' ensures that the variable 'num' will be randomized with all possible values (0 to 15) before it repeats. It avoids repeating the same value until all combinations are used.*

Q491. What is the output when randomizing a bit vector 'data' with the constraint that the sum of all bits should be 3?

```
class RandomExample;
    rand bit [7:0] data;
    constraint sum_of_bits { $countones(data) == 3; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0b", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized data: A value with exactly 3 ones in its binary representation.*

Q492. What will be the randomized output for 'data' when the following constraints are applied?

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data > 50 && data mod 4 == 0; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized data: A number greater than 50 that is divisible by 4.*

Q493. What is the result of randomizing the 'data' with a constraint where the value should be either 20 or 50?

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data == 20 || data == 50; }
```

```
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized data: Either 20 or 50.*

Q494. What will happen when applying the following constraint on ‘data‘?

```
class RandomExample;
    rand bit [3:0] data;
    constraint valid_data { data inside { 4, 8, 12 }; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized data: One of the values 4, 8, or 12.*

Q495. What will be the output of the following code when ‘num‘ is randomized?

```
class RandomExample;
    rand bit [7:0] num;
    constraint valid_data { num >= 10 && num <= 50 && num mod 2 == 0; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized num: %0d", re.num);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized num: A number between 10 and 50 that is even.*

Q496. How will the randomization behave if we apply the following constraint on a variable?

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data inside {5, 10, 15, 20, 25}; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *Randomized data: One of the values 5, 10, 15, 20, or 25.*

Q497. What happens when the following randomization pattern is applied?

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data dist { 0: 50, 1: 30, 2: 20 }; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
    else
        $display("Randomization failed");
end
```

Answer: *The randomization will give '0' with a probability of 50%, '1' with 30%, and '2' with 20%.*

Q498. Predict the output when using the following constraint pattern:

```
class RandomExample;
    rand bit [7:0] data;
    constraint valid_data { data dist { 0:1, 1:1, 2:1, 3:2 }; }
endclass

RandomExample re = new;
initial begin
    if (re.randomize())
        $display("Randomized data: %0d", re.data);
```

```
        else
            $display("Randomization failed");
    end
```

Answer: *The randomization will give '3' with the highest probability, followed by '0', '1', and '2'.*

Q521. Write a SystemVerilog program to print the following inverted star triangle:

```
* * * * *
* * * *
* * *
* *
*
```

Hint: Adjust the spaces before the stars in each row for the inverted triangle.

Q522. Write a SystemVerilog program to print the following number triangle with decreasing numbers:

```
5 4 3 2 1
5 4 3 2
5 4 3
5 4
5
```

Hint: Use nested loops to control both the rows and the decreasing numbers.

Q523. Write a SystemVerilog program to print the following pattern:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Hint: Use a counter to print increasing numbers in each row.

Q524. Write a SystemVerilog program to print the following half-diamond pattern of stars:

```
 *
 * *
 * * *
 * * * *
 * * *
 * *
 *
```

Hint: Combine increasing and decreasing loops to generate the half-diamond.

Q525. Write a SystemVerilog program to print the following number pattern:

```
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

Hint: Use nested loops to print the same number multiple times per row.

Q526. Write a SystemVerilog program to print a number square in which the numbers are arranged in a diagonal fashion:

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

Hint: Use two loops to control the row and column, with 1s on the diagonal.

Q527. Write a SystemVerilog program to print a number pyramid with the following sequence:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Hint: Use nested loops and a counter to print the sequence in a pyramid shape.

Q528. Write a SystemVerilog program to print the following hollow right-angle triangle:

```
*
* *
*   *
*     *
* * * * *
```

Hint: Print stars on the borders and spaces in the interior.

Q529. Write a SystemVerilog program to print the following mirrored half-diamond number pattern:

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Hint: Use nested loops for increasing and decreasing the rows and numbers.

Q530. Write a SystemVerilog program to print the following pattern:

```
* 1 * 2 * 3 *
* 4 * 5 * 6 *
* 7 * 8 * 9 *
```

Hint: Alternate between stars and numbers across all rows.

Q531. Write a SystemVerilog program to print a square pattern with increasing numbers along rows:

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

Hint: Use nested loops to print numbers in a square format.

Q532. Write a SystemVerilog program to print the following number triangle with increasing numbers:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Hint: Use a counter to control the number of numbers printed in each row.

Q533. Write a SystemVerilog program to print the following hollow diamond pattern:

```
    *
   * *
  *   *
 *     *
*       *
 *     *
  *   *
   * *
    *
```

Hint: Use spaces to create the hollow portion of the diamond.

Q534. Write a SystemVerilog program to print the following pattern with alternating stars and numbers:

```
* 1 * 2 * 3
* 4 * 5 * 6
* 7 * 8 * 9
* 10 * 11 * 12
```

Hint: Alternate stars and numbers using loops.

Q535. Write a SystemVerilog program to print a reverse triangle pattern of stars:

```
* * * * *
 * * * *
  * * *
   * *
    *
```

Hint: Adjust the number of stars printed in each row to form a reverse triangle.

Q536. Write a SystemVerilog program to print the following number sequence:

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Hint: Use loops to decrease the number of elements printed in each row.

Q537. Write a SystemVerilog program to print a hollow square of stars:

```
* * * *
*     *
*     *
* * * *
```

Hint: Print stars at the borders of the square and spaces in the interior.

Q538. Write a SystemVerilog program to print the following half-diamond pattern of numbers:

```
1
2 3
4 5 6
7 8 9 10
7 8 9
4 5 6
2 3
1
```

Hint: Use loops for both increasing and decreasing the numbers.

Q539. Write a SystemVerilog program to print a reverse pyramid of stars:

```
* * * * *
 * * * *
  * * *
   * *
    *
```

Hint: Adjust the spaces before printing stars in each row to form a reverse pyramid.

Q540. Write a SystemVerilog program to print the following inverted number pattern:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Hint: Use nested loops to print the numbers in decreasing order.

Q541. Write a SystemVerilog program to print the following spiral number pattern:

```
1 2 3
8 9 4
7 6 5
```

Hint: Use loops to manage the spiral pattern.

Q542. Write a SystemVerilog program to print a hollow right-angle triangle pattern of stars:

```
*
* *
*   *
*     *
* * * * *
```

Hint: Handle both spaces and stars correctly using nested loops.

Q543. You are given a random variable `int a` with a constraint that it must be between 1 and 10 (inclusive). Write a SystemVerilog code to randomize the value of `a` so that it is either 3, 5, or 7. Ensure that the value of `a` cannot be randomized to any other value outside this set.

Hint: Use a constraint with a specific set of allowed values.

Q544. Write a SystemVerilog program to generate a random 3-bit binary number such that the number has exactly two ones (i.e., it is a 3-bit number with exactly two ones). Use randomization with constraints to achieve this.

Hint: Define a 3-bit number and add constraints to ensure exactly two 1's.

Q545. Write a SystemVerilog code to generate a random 4-digit number such that the sum of the digits is equal to 10. The randomization should only generate numbers with valid digits (0-9) and satisfy this condition.

Hint: Use a constraint to sum the digits and ensure they are valid.

Q546. In a lottery game, the ticket number is generated randomly between 1 and 100. Write a SystemVerilog program to randomize the ticket number and then constrain it so that the winning numbers are limited to 1, 3, 5, 7, 9, 15, 20, 25, 30, 35. Any other number should be excluded.

Hint: Use a set constraint to limit the possible random values.

Q547. Create a SystemVerilog program that generates a random number `x` in the range of 1 to 100. Then, add a constraint to the value of `x` such that it should always be even. Ensure that the random value is constrained to be even and within the specified range.

Hint: Use the modulus operator for even constraint.

Q548. You are given a random 5-bit number `rand_num[4:0]`. Write a SystemVerilog constraint to generate a number such that the number is always a multiple of 4 and is less than 20. This means that the possible values for `rand_num` can be 0, 4, 8, 12, or 16.

Hint: Use a modulo constraint to constrain the number to multiples of 4.

Q549. You need to generate a random array of size 10 where each element is constrained to be either 5 or 10. However, there must be exactly 3 elements that are equal to 5 and 7 elements that are equal to 10. Write a SystemVerilog constraint that ensures this distribution.

Hint: Use an array constraint with a specific count for 5's and 10's.

Q550. Write a SystemVerilog program that generates a random 6-bit number. The number should be constrained such that the number of 1's in the 6-bit binary number is always greater than or equal to 3.

Hint: Use a constraint to count the number of ones in the binary number and enforce the condition.

Q551. Create a random string of 8 characters where each character is a lowercase alphabet (a-z). Additionally, ensure that the string must contain at least 2 vowels (a, e, i, o, u). Write the SystemVerilog code for this randomization problem.

Hint: Use randomization constraints to pick characters and enforce the vowel count.

Q552. You are tasked with generating a random number `x` from 1 to 100. The number should always be a multiple of 5 but never greater than 50. Write a SystemVerilog constraint to achieve this.

Hint: Use constraints to restrict values to multiples of 5 within the range.

Q553. Write a SystemVerilog code to generate a random array of 4 elements. The values of the array should be between 1 and 10, but the sum of the values must be 25.

Hint: Use constraints to generate values and restrict their sum.

Q554. Create a random 8-bit number such that the number is divisible by 3. Additionally, ensure that the 8-bit number must have an even number of 1's. Write the SystemVerilog code for this constraint-based randomization.

Hint: Use constraints for divisibility and count of 1's in the number.

Q555. Write a SystemVerilog program to randomize a 4-bit array such that it represents a binary number divisible by both 2 and 3. What constraint would you apply to achieve this?

Hint: The number should be divisible by 6 (i.e., divisible by both 2 and 3).

Q556. You need to generate a 4-bit random number that represents an even number, but the sum of the digits (binary 1's) must be exactly 2. How would you do this using SystemVerilog?

Hint: Use constraints for even numbers and the number of 1's in the number.

Q557. Write a SystemVerilog program to generate a random 5-bit number, but constrain the number such that the number of 1's in the binary number must be an even number.

Hint: Use a constraint to count the number of 1's and enforce the evenness condition.

Q558. Generate a random 2-dimensional array of size 3x3, where each element is a number between 1 and 100. Write a SystemVerilog program that constrains the array such that the sum of all the numbers in the array is exactly 150.

Hint: Use constraints to control both the values and the sum of the array elements.

Q559. Write a SystemVerilog code to randomize a 4-bit number such that the number must be a power of 2 (e.g., 1, 2, 4, 8, 16). Additionally, the number must be greater than or equal to 8. Write the appropriate constraint for this.

Hint: Use constraints to ensure the value is both a power of 2 and within the specified range.

Q560. Write a SystemVerilog program to generate a random 3-bit number, but constrain the number such that it is not divisible by 3.

Hint: Use a modulus constraint to ensure the number is not divisible by 3.

Q561. Write a SystemVerilog program to generate a random 4-bit number such that it is not divisible by 5. Additionally, it should not have more than 2 ones in its binary representation.

Hint: Use modulus constraints for divisibility and count the ones in the binary representation.

Q562. You are tasked with generating a random 8-bit number, but it must be a palindrome in binary. That is, the number should read the same forward and backward (e.g., 10101010). Write the SystemVerilog code to achieve this randomization.

Hint: Use symmetry constraints to enforce the palindrome condition.

Q563. Write a SystemVerilog code to generate a random 5-bit number that is divisible by 4 but not divisible by 6. Ensure the number falls within the 0 to 31 range.

Hint: Use a combination of modulus constraints to check divisibility by 4 and 6.

Q564. Generate a random 6-bit number with exactly 3 zeros and 3 ones. Ensure that the number is a valid binary representation.

Hint: Use a constraint to set the number of 1's and 0's.

Q565. Write a SystemVerilog program that generates a random 3-bit number, but it must be greater than or equal to 4 and a multiple of 2. What constraints would you apply to achieve this?

Hint: Use constraints for greater-than-or-equal and divisibility.

Q566. You are given a random array of integers `arr[5:0]`. Write a constraint such that at least two elements in the array must be even and the rest can be any value between 1 and 10.

Hint: Use a count constraint to enforce at least two even numbers in the array.

Q567. Create a random string of 6 characters where each character is a lowercase letter. Write a constraint to ensure that there is at least one vowel in the string.

Hint: Use a constraint to count the vowels in the string.

Q568. Write a SystemVerilog code to generate a random array of 4 integers, where each integer is between 5 and 20, but the sum of the array elements must be exactly 50.

Hint: Use a sum constraint to ensure the array elements add up to 50.

Q569. Write a SystemVerilog program to generate a random 8-bit binary number such that the number must be a power of 2. Additionally, constrain the value to be less than or equal to 64.

Hint: Use a constraint for powers of 2 and limit the range.

Q570. You are tasked with generating a 4-bit number where the number of 1's should always be less than the number of 0's. Write the SystemVerilog code for this constraint-based randomization.

Hint: Count the 1's and 0's and apply a constraint for the number of 1's.

Q571. Create a random 5-bit number such that the number of 1's is greater than or equal to 3. Ensure the number is always even.

Hint: Use constraints for the number of 1's and evenness.

Q572. Write a SystemVerilog code to generate a random 3-bit number where the number is divisible by 3 and less than 6.

Hint: Use constraints for divisibility and range.

Q573. You are given a random variable `int a`. Write a constraint to randomize the value of `a` such that it is always greater than or equal to 20 but less than 50 and divisible by 5.

Hint: Apply a combination of range and divisibility constraints.

Q574. Generate a random number `x` from 1 to 100. Then, constrain it to be a prime number. Write the SystemVerilog code to achieve this.

Hint: Use a constraint to enforce prime number conditions.

Q575. Write a SystemVerilog code to generate a random number between 1 and 10. The number should be a perfect square. What constraints would you apply to achieve this?

Hint: Use a constraint for perfect squares.

Q576. Write a SystemVerilog program that generates a random array of size 4, where each element is between 1 and 5. The array should be constrained such that at least two elements are equal to 4.

Hint: Use a constraint to enforce that at least two elements are equal to 4.

Q577. Create a random string of 10 characters where each character is a digit between 0 and 9. Write a constraint to ensure that the string contains at least two consecutive 1's.

Hint: Use string manipulation and regular expression-style constraints for the consecutive digits.

Q578. Write a SystemVerilog program that generates a random 4-bit number, but the number must contain at least 3 ones in its binary representation. Additionally, ensure that the number is a multiple of 4.

Hint: Use constraints for both the number of ones and divisibility.

Q579. You are tasked with generating a random number `x` between 1 and 100, but the value of `x` must be a multiple of 3 and a multiple of 5. Write a SystemVerilog constraint to achieve this.

Hint: Use the least common multiple (LCM) or direct modulus constraints.

Q580. Write a SystemVerilog program to generate a random 6-bit number such that it is divisible by 3 and contains at least 2 ones in the binary representation.

Hint: Apply constraints for divisibility by 3 and the number of ones.

Q581. Write a SystemVerilog code to generate a random number `a` such that it is always greater than or equal to 10 and less than 30. Additionally, the number `a` should be a multiple of 4.

Hint: Use a combination of range and divisibility constraints.

Q582. You are tasked with generating a random 8-bit binary number. The number must be divisible by 6, and the sum of the digits (binary 1's) should be exactly 4.

Hint: Use divisibility constraints and count the number of 1's in the binary number.

Q583. Generate a random number `a` between 1 and 100. Then constrain `a` to be divisible by 2 or 5 but not both.

Hint: Use a combination of modulus constraints and logical operators.

Q584. Write a SystemVerilog program to generate a random array of size 3. Each element should be a unique number between 1 and 5. Ensure the values are distinct.

Hint: Use a uniqueness constraint.

Q585. Write a SystemVerilog code to generate a random number `n` between 1 and 100. The number must not be divisible by 2, 3, or 5.

Hint: Use a combination of modulus constraints to enforce the divisibility conditions.

Q586. Write a SystemVerilog program that generates a random 3-bit number such that it must be less than 6 and odd. Apply appropriate constraints to ensure this condition is met.

Hint: Apply range and odd-number constraints.

Q587. Generate a random 4-bit number such that the number of 1's in the binary representation is exactly 2, and the number should be a multiple of 2.

Hint: Use constraints to count 1's and enforce divisibility by 2.

Q588. Write a SystemVerilog program to generate a random string of 5 characters. The string should only contain vowels (a, e, i, o, u), and the string should contain at least 3 vowels.

Hint: Use a constraint to count the vowels in the string.

Q589. Write a SystemVerilog code to generate a random array of 6 integers. Each integer should be between 10 and 20, and the sum of the integers must be 100.

Hint: Use constraints for range and sum of the array elements.

Q590. Write a SystemVerilog program that generates a random number `n` where `n` is between 1 and 10. The number should always be a prime number.

Hint: Apply a constraint to restrict the values to prime numbers only.

Q591. Write a SystemVerilog program to generate a random 6-bit number where the number of ones must be even.

Hint: Use a constraint to count the number of ones in the binary number and enforce the even condition.

Q592. You are tasked with generating a 7-bit number. The number must be divisible by 5 and have an odd number of ones in its binary representation.

Hint: Use divisibility constraints for 5 and count the number of ones in the number.

Q593. Write a SystemVerilog program that generates a random 3-bit number where the number of 1's must be greater than or equal to 2. Additionally, the number must be an even number.

Hint: Apply a constraint for the number of 1's and enforce evenness.

Q594. Write a SystemVerilog code to generate a random 4-bit number such that the number is divisible by both 2 and 3.

Hint: Use divisibility constraints for both 2 and 3.

Q595. Create a random 6-bit number, ensuring that the number is divisible by 4 and contains no more than 3 ones in its binary representation.

Hint: Use divisibility constraints for 4 and a constraint for the number of ones.

Q596. You are tasked with generating a random string of 5 characters. The string should consist only of consonants, and at least 2 characters in the string must be vowels.

Hint: Use randomization constraints for consonants and enforce the vowel count.

Q597. Write a SystemVerilog program to generate a random 4-bit number such that the number has an odd number of 1's in its binary representation and is a multiple of 3.

Hint: Apply a constraint for oddness in the number of ones and divisibility by 3.

Q598. Create a random number between 1 and 10, ensuring that the number is a perfect square and divisible by 3.

Hint: Apply constraints for perfect squares and divisibility by 3

Q599. Write a SystemVerilog program that generates a random array of 4 elements, each element being a random number between 0 and 50. The sum of the elements must be greater than or equal to 100.

Hint: Apply a sum constraint.

Q600. Write a SystemVerilog program to generate a random 5-bit number, but the number must not be divisible by 5 and must contain an even number of ones in its binary representation.

Hint: Use a modulus constraint for divisibility by 5 and a constraint for the number of ones.

Q601. Write a SystemVerilog program to generate a random array of 5 integers, each between 10 and 20. Ensure that the sum of the array elements is between 50 and 70.

Hint: Use constraints for both the range and the sum of the array.

Q602. Write a SystemVerilog program to generate a random 3D array of integers. Each element should be between 0 and 100. Add constraints such that the sum of each row in the 3D array should be greater than 100.

Hint: Apply row sum constraints using loops over the 3D array.

Q603. Write a SystemVerilog code to generate a random array of size 6 where each element is a number between 1 and 10. Additionally, the sum of the array elements should be exactly 30.

Hint: Use a sum constraint for the array elements.

Q604. You are tasked with generating a random 5-element array of integers, each between 0 and 20. The sum of the array elements must be an even number.

Hint: Use a sum constraint and ensure it is even.

Q605. Generate a random array of 10 integers, each between 0 and 100. Apply constraints such that no two adjacent elements can be equal.

Hint: Use a constraint to enforce adjacent element inequality.

Q606. Create a random 7-element array of integers where each element should be between 5 and 15. Ensure that at least 3 elements in the array are equal to 10.

Hint: Use a constraint to enforce the number of elements equal to 10.

Q607. Write a SystemVerilog program to generate a random array of 4 boolean values. The array should be constrained such that exactly two values are true.

Hint: Use a constraint to enforce the exact number of true values.

Q608. Write a SystemVerilog code to generate a random 3D array of integers of size [4:0][3:0][5:0]. Ensure that the sum of each 2D slice (4x5) in the 3D array is greater than 50.

Hint: Loop over the 2D slices in the 3D array and apply sum constraints.

Q609. Write a SystemVerilog code to generate a random array of 8 integers between 1 and 10. The array must be sorted in ascending order after randomization.

Hint: Use a post-randomization sorting method or constraint.

Q610. You are tasked with generating a random 5-element array of integers, where the first and last elements are both constrained to be even numbers. The other elements can be any number between 1 and 10.

Hint: Apply constraints for the first and last elements to be even.

Q611. Write a SystemVerilog program to generate a random 6-element array where each element is a random floating-point number between 0.0 and 1.0. Apply a constraint to ensure the average of the array elements is greater than 0.5.

Hint: Use a sum constraint for the floating-point numbers.

Q612. Generate a random array of 10 integers, each between 1 and 20. The array should contain no more than 3 elements equal to 10.

Hint: Use a constraint to limit the number of 10's in the array.

Q613. Write a SystemVerilog code to generate a random array of 7 integers, each between 10 and 30. The array should contain at least 2 elements equal to 20.

Hint: Use a constraint to enforce the number of elements equal to 20.

Q614. Write a SystemVerilog program that generates a random 5-element array, where each element is between 1 and 10, but the array must be sorted in descending order.

Hint: Use a constraint to enforce descending order.

Q615. Create a random array of size 4 with each element between 1 and 20. The array should be constrained such that at least one element is greater than 15 and at least one element is less than 5.

Hint: Use a constraint to enforce the range conditions.

Q616. You are tasked with generating a random 2D array of size [4:0][3:0] where each element is a random number between 10 and 50. Ensure that all the elements in each row are unique.

Hint: Use a uniqueness constraint for each row.

Q617. Write a SystemVerilog code to generate a random array of 6 elements where each element is between 0 and 9. Apply a constraint that ensures that no element in the array is repeated.

Hint: Use a uniqueness constraint to ensure no repetition.

Q618. Write a SystemVerilog program to generate a random 5-element array of integers between 1 and 100. The array should be constrained such that the sum of the array is always a prime number.

Hint: Use a constraint to check for prime sums.

Q619. Create a random array of 4 boolean values. The constraint should ensure that at least two values in the array are false.

Hint: Use a constraint to enforce the number of false values.

Q620. Write a SystemVerilog code to generate a random array of 10 elements, each being a floating-point number between 0.0 and 5.0. The array must contain at least 3 elements greater than 4.0.

Hint: Use a constraint to enforce the number of elements greater than 4.0.

Q621. You are tasked with generating a random 5-element array, where each element is a floating-point number between 0.0 and 1.0. Ensure that the average of the array elements is exactly 0.5.

Hint: Use a sum constraint to ensure the average is exactly 0.5.

Q622. Write a SystemVerilog program that generates a random 3-element array where each element is constrained to be prime numbers between 2 and 20.

Hint: Apply a constraint to restrict the elements to prime numbers.

Q623. Create a random array of 4 integers between 1 and 20, where each element must be distinct. The sum of the array elements should be exactly 30.

Hint: Use uniqueness and sum constraints.

Q624. Write a SystemVerilog code to generate a random 6-element array where the sum of the array is less than or equal to 50. Each element must be between 1 and 10.

Hint: Apply a sum constraint and a range constraint.

Q625. Generate a random array of 8 elements, each between 0 and 50. Ensure that the array is sorted in ascending order.

Hint: Use a post-randomization sorting method or constraint.

Q626. Write a SystemVerilog program to generate a random 7-element array of integers between 5 and 25. Apply a constraint that the sum of the array elements must be divisible by 5.

Hint: Use a sum constraint and divisibility check.

Q627. Create a random 3D array of integers where the size of the array is [3:0][3:0][3:0]. Ensure that the sum of each 2D slice is greater than 50.

Hint: Loop through the 2D slices and apply sum constraints.

Q628. Write a SystemVerilog program to generate a random array of size 6, with values between 1 and 100. Apply a constraint such that no element is greater than 50.

Hint: Use a range constraint for each array element.

Q629. You are tasked with generating a random 5-element array where each element is between 1 and 15. Apply constraints to ensure that the average value of the array is between 5 and 10.

Hint: Use a sum constraint to control the average.

Q631. Write a SystemVerilog code to generate a random 2D array of integers of size [3:0][3:0], where all values are between 1 and 50, and the diagonal elements must be equal.

Hint: Add a constraint so that array[i][i] == const across all rows/columns.

Q632. Generate a random 5-element array of integers such that each element is a Fibonacci number less than 100.

Hint: Use pre-defined Fibonacci array and constrain selection.

Q633. Write a SystemVerilog program that generates a random array of 7 integers between 1 and 20. Ensure that no value appears more than twice.

Hint: Use repetition count constraints.

Q634. Generate a random 1D array of 10 integers such that elements at even indices are even, and at odd indices are odd.

Hint: Use a looped constraint based on index parity.

Q635. Create a random array of size 6, with values between 1 and 30. Apply a constraint that elements at index 0 and 5 must be greater than all other elements.

Hint: Add element[i] ¡ element[0] element[5] for i in 1 to 4.

Q636. Write a SystemVerilog program that generates a 2D array [4][4], where each row is a rotated version of the previous one.

Hint: Use circular shift logic with constraints.

Q637. Generate a random array of 5 integers such that the maximum value occurs exactly once.

Hint: Constrain uniqueness of maximum.

Q638. Write a constraint to create a random 10-element array where no three consecutive elements form an arithmetic progression.

Hint: Use ‘array[i+2] - array[i+1] != array[i+1] - array[i]‘.

Q639. Write a SystemVerilog class that generates a random 5-element array such that the sum of the first three elements equals the sum of the last two.

Hint: Constraint: ‘a[0]+a[1]+a[2] == a[3]+a[4]‘.

Q640. Create a 4-element array such that the XOR of all elements is a given constant value.

Hint: Use XOR reduction and constrain result.

Q641. Write a program that generates a random array of 6 integers where the first half and second half of the array have equal sums.

Hint: Constraint: ‘a[0]+a[1]+a[2] == a[3]+a[4]+a[5]‘.

Q642. Generate a random 7-element array with values between 1 and 15, and ensure that there are no local minima (i.e., no element is smaller than both its neighbors).

Hint: For each i, ensure ‘a[i] > a[i-1] —— a[i] > a[i+1]‘.

Q643. Write a constraint to generate an array such that every second element is greater than the first and less than the third (like a peak in the middle).

Hint: Use conditional chaining constraints.

Q644. Generate a random array of size 5 with all elements divisible by both 2 and 3.

Hint: Use ‘mod‘ constraint: ‘a[i]

Q645. Create an array of 6 elements where the elements form a geometric progression with a random ratio between 2 and 5.

Hint: Use multiplicative relationship with a random ratio.

Q646. Write SystemVerilog code to generate a random array of size 8 where the difference between any two adjacent elements is not more than 3.

Hint: Use ‘abs(a[i] - a[i+1]) ¡= 3‘.

Q647. Generate a random array of size 5 that contains a strictly increasing sequence.

Hint: Constraint: ‘a[i] < a[i+1]‘ for i in 0 to 3.

Q648. Write a SystemVerilog constraint to create a palindromic array of size 6.

Hint: Constraint: ‘a[i] == a[5-i]‘.

Q649. Create a random 2D array [3][3] with values such that the sum of all rows and all columns are equal.

Hint: Constraint: Row sums == Column sums.

Q650. Generate a random array of 5 elements such that all elements are square numbers (e.g., 1, 4, 9, ...) and each is unique.

Hint: Use a fixed array of square numbers and constrain using ‘dist‘ or ‘choose‘.

Q651. What is the purpose of the ‘static‘ keyword in SystemVerilog? Explain with a small code snippet.

Q652. Write a program to demonstrate the difference between a static and a non-static variable inside a class method.

Q653. Can a static variable be randomized in SystemVerilog? Justify your answer with an example.

Q654. Create a class with a static counter variable that increments every time a new object is created. Show the value of the counter after creating 3 objects.

Q655. Show a scenario where static variables can cause issues in UVM or parallel simulations.

Q656. Explain how a static method differs from an instance method. Provide code to call both types from the testbench.

Q657. Can a static method access non-static variables? Provide a reason and an example.

Q658. Write a code to share a static array among multiple objects. Show how changing it in one affects the others.

Q659. What is the visibility scope of a static variable declared inside a function?

Q660. Create a simulation showing how a static class variable retains its value across simulation steps.

Q661. Define the ‘const‘ keyword in SystemVerilog. How is it different from ‘localparam‘?

Q662. Can a ‘const‘ variable be initialized inside a constructor? Demonstrate with an example.

Q663. Create a class with a ‘const‘ member variable and show how you would initialize it with ‘new()‘.

Q664. What happens if you try to assign a new value to a ‘const‘ variable after declaration? Prove it with a simulation.

Q665. Compare and contrast ‘parameter‘, ‘localparam‘, and ‘const‘ in terms of usage and scope.

Q666. Create a const 2D array and demonstrate its read-only nature with an attempted write.

Q667. Can you randomize a ‘const‘ variable? Explain your reasoning and show the compile-time error if any.

Q668. What is the benefit of using ‘const‘ in testbench components?

Q669. How does the ‘const‘ keyword improve code readability and reduce bugs in large UVM testbenches?

Q670. Create a struct with a ‘const‘ field. Try accessing and modifying it—what do you observe?

Q671. What is the role of the ‘virtual‘ keyword in SystemVerilog?

Q672. Create a base class with a virtual method and override it in a derived class. Show polymorphic behavior.

Q673. What is a virtual interface? How does it differ from a regular interface?

Q674. Demonstrate how a virtual method enables dynamic dispatch at runtime.

Q675. Can we declare a class as virtual in SystemVerilog? What does that imply?

Q676. Write a testbench where a virtual interface is used to abstract the DUT signal connectivity.

Q677. Can we override a virtual function with a non-virtual function? Show what the simulator does.

Q678. Demonstrate late binding using a virtual method in base and child class, with a base handle.

Q679. Explain the usage of ‘virtual task‘ vs ‘virtual function‘. When should one be used over the other?

Q680. Show how virtual interfaces help in stimulus generation in constrained environments.

Q681. What are some best practices when using static variables inside reusable classes?

Q682. Why might overusing static variables break simulation scalability?

Q683. Create a virtual base class ‘transaction‘ and derived class ‘read_txn‘ and ‘write_txn‘. Add a static counter for each transaction type created.

Q684. Can you use ‘const‘ with ‘static‘ together in a class? Provide a valid example.

Q685. What issues arise if a ‘virtual‘ function accesses a ‘static‘ class member? Simulate and explain.

Q686. Write a virtual class ‘Packet‘, then implement ‘TCPPacket‘ and ‘UDPPacket‘. Add a ‘static‘ variable to count how many total packets are sent.

Q687. In SystemVerilog, can ‘static‘ methods be overridden using ‘virtual‘? Why or why not?

Q688. Demonstrate that ‘const‘ variables must be initialized exactly once using a constructor.

Q689. Can a ‘const‘ variable in a class be initialized via randomization? Why or why not?

Q690. Show an example of abstracting an environment using a ‘virtual interface‘ in a modular UVM testbench.

Q691. Create a parent class with a static variable and access it from a child class. Show inheritance of static members.

Q692. Create a code snippet where a virtual method selects a type of report based on a ‘const‘ configuration variable.

Q693. Can a ‘const‘ variable be assigned via function return values? Demonstrate with legal and illegal cases.

Q694. Discuss how combining ‘const‘ and ‘static‘ makes a variable similar to a constant global setting.

Q695. Write a base class ‘Shape‘ with a virtual method ‘area()‘. Create ‘Circle‘ and ‘Square‘ subclasses and override ‘area()‘.

Q696. Create a simulation where a ‘static const‘ configuration is used inside a factory pattern to determine object creation.

Q697. Is it valid to declare a ‘virtual‘ function as ‘static‘? Why or why not? Provide code and error.

Q698. Create a scenario where a ‘const‘ parameter controls a randomized field’s max value. Show compile-time vs run-time behavior.

Q699. Why can’t you declare a virtual method in a non-class scope? Explain with valid context.

Q700. Compare ‘static‘, ‘const‘, and ‘virtual‘ keywords with respect to their scope, lifetime, and behavior in OOP-based SystemVerilog.

Q701. Predict the output:

```
class A;
  const int x = 5;
  function void print();
    $display("x = %0d", x);
  endfunction
endclass

module tb;
  A obj = new;
  initial obj.print();
endmodule
```

Q702. What will happen if the following is run?

```
class A;
  const int y;
  function new();
    y = 10;
  endfunction
```

```
endclass

module tb;
  A obj = new;
  initial $display("y = %0d", obj.y);
endmodule
```

Q703. Predict the output:

```
class A;
  const int x = 7;
  function void modify();
    //x = 8; // Uncomment to test
    $display("x = %0d", x);
  endfunction
endclass

module tb;
  A a = new;
  initial a.modify();
endmodule
```

Q704. Predict the output:

```
class Counter;
  static int count = 0;
  function new();
    count++;
  endfunction
endclass

module tb;
  initial begin
    Counter a = new;
    Counter b = new;
    $display("Count = %0d", Counter::count);
  end
endmodule
```

Q705. Static variable accessed from two objects:

```
class Test;
  static int value = 20;
  function void modify();
    value += 5;
  endfunction
endclass

module tb;
  Test t1 = new, t2 = new;
  initial begin
```

```
    t1.modify();
    t2.modify();
    $display("Value = %0d", Test::value);
  end
endmodule
```

Q706. Predict output:

```
class A;
  int i = 2;
  static int j = 4;
  function void print();
    $display("i=%0d, j=%0d", i, j);
  endfunction
endclass

module tb;
  A a = new;
  a.i = 10;
  A.j = 20;
  initial a.print();
endmodule
```

Q707. Virtual dispatch:

```
class Base;
  virtual function void print();
    $display("Base");
  endfunction
endclass

class Derived extends Base;
  function void print();
    $display("Derived");
  endfunction
endclass

module tb;
  Base obj;
  initial begin
    obj = new Derived();
    obj.print();
  end
endmodule
```

Q708. No virtual keyword:

```
class Base;
  function void print();
    $display("Base");
  endfunction
```

```
endclass

class Derived extends Base;
  function void print();
    $display("Derived");
  endfunction
endclass

module tb;
  Base obj = new Derived();
  initial obj.print();
endmodule
```

Q709. Mixing virtual and static:

```
class A;
  static int val = 3;
  virtual function void display();
    $display("Value: %0d", val);
  endfunction
endclass

class B extends A;
  function void display();
    val = val + 7;
    $display("Updated: %0d", val);
  endfunction
endclass

module tb;
  A obj = new B();
  initial begin
    obj.display();
    obj.display();
  end
endmodule
```

Q710. Predict output:

```
class Outer;
  class Inner;
    function void show();
      $display("Inside Inner");
    endfunction
  endclass
endclass

module tb;
  Outer::Inner obj = new;
  initial obj.show();
```

```
endmodule
```

Q711. Nested access with outer handle:

```
class Outer;
  int val = 9;
  class Inner;
    function void show();
      $display("Accessed Inner");
    endfunction
  endclass
endclass

module tb;
  Outer o = new;
  Outer::Inner i = new;
  initial begin
    $display("Outer.val = %0d", o.val);
    i.show();
  end
endmodule
```

Q712. Static method in nested class:

```
class Parent;
  class Helper;
    static function void msg();
      $display("Hello from nested static method");
    endfunction
  endclass
endclass

module tb;
  initial Parent::Helper::msg();
endmodule
```

Q713. Predict the output:

```
class A;
  static int count = 0;
  function new();
    count++;
  endfunction
  virtual function void show();
    $display("Count = %0d", count);
  endfunction
endclass

class B extends A;
  function void show();
    $display("Derived Count = %0d", count*2);
```

```
  endfunction
endclass

module tb;
  A obj = new B();
  initial obj.show();
endmodule
```

Q714. What's printed?

```
class Top;
  const int val = 10;
  class Inner;
    static function void test();
      $display("Testing");
    endfunction
  endclass
endclass

module tb;
  initial Top::Inner::test();
endmodule
```

Q750. Explain what will be printed here:

```
class A;
  static int x = 5;
  virtual function void display();
    $display("x = %0d", x);
  endfunction
endclass

class B extends A;
  function void display();
    x = x + 2;
    $display("x = %0d", x);
  endfunction
endclass

module tb;
  A obj = new B();
  initial begin
    obj.display();
    obj.display();
  end
endmodule
```

Q751. Define a packed struct named 'Packet' with fields: 'bit [3:0] id', 'bit [7:0] data'. Display its size in bits.

Q752. What is the output?

```
typedef struct packed {
  bit [3:0] header;
  bit [7:0] payload;
} pkt_t;

module tb;
  pkt_t p = '{4'b1010, 8'hFF};
  initial $display("%b", p);
endmodule
```

Q753. What's the key difference between 'packed' and 'unpacked' struct in terms of memory layout and simulation?

Q754. Create a struct for student record using typedef having fields: name[20], age (int), roll$_no(int).Isthispacked$?

Q755. Predict the output:

```
typedef struct {
  int a;
  bit [3:0] b;
} myStruct;

module tb;
  myStruct s;
  initial begin
    s.a = 10;
    s.b = 4'b1010;
    $display("%0d, %b", s.a, s.b);
  end
endmodule
```

Q756. Can a struct contain a union? Provide a valid example.

Q757. Create a packed union containing two members: one 16-bit integer and another as 4 x 4-bit bit-fields.

Q758. What is the output?

```
typedef union packed {
  bit [15:0] word;
  struct packed {
    bit [7:0] lower;
    bit [7:0] upper;
  } bytes;
} data_u;

module tb;
  data_u u;
  initial begin
    u.word = 16'hABCD;
    $display("Lower: %h, Upper: %h", u.bytes.lower, u.bytes.upper);
  end
```

```
endmodule
```

Q759. Predict output:

```
typedef union packed {
  int a;
  bit [31:0] b;
} u_type;

module tb;
  u_type u;
  initial begin
    u.a = 32'hFFFFFFFF;
    $display("b = %0d", u.b);
  end
endmodule
```

Q760. Write a packed struct with a union inside that holds either an address (32-bit) or data (16-bit).

Q761. Predict result:

```
typedef struct packed {
  bit [3:0] opcode;
  union packed {
    bit [7:0] imm;
    bit [7:0] addr;
  } payload;
} inst_t;

module tb;
  inst_t inst = '{4'b1100, 8'hAB};
  initial $display("Full: %b", inst);
endmodule
```

Q762. Use ‘typedef‘ to define an alias for a struct that holds x, y coordinates as int.

Q763. What's the output?

```
typedef struct {
  int x;
  int y;
} point_t;

module tb;
  point_t p = '{10, 20};
  initial $display("(%0d, %0d)", p.x, p.y);
endmodule
```

Q764. Can we typedef a union? Show how and explain benefits.

Q765. Predict result:

```
typedef struct {
```

```
  byte a;
  int b;
} s1;

typedef struct packed {
  byte a;
  int b;
} s2;

module tb;
  initial begin
    $display("Size s1 = %0d", $bits(s1));
    $display("Size s2 = %0d", $bits(s2));
  end
endmodule
```

Q766. Why is ‘packed‘ preferred for synthesis and verification?

Q767. Show how to slice a packed struct bit-wise:

```
typedef struct packed {
  bit [3:0] header;
  bit [11:0] data;
} frame_t;

module tb;
  frame_t f = '{4'b1010, 12'hABC};
  initial $display("Bit 7:4 = %b", f[7:4]);
endmodule
```

Q768. Create a struct within a struct using typedef. Use values and print nested fields.

Q769. Predict output:

```
typedef struct {
  bit [3:0] opcode;
  struct packed {
    bit [7:0] addr;
    bit [7:0] data;
  } memory;
} mem_t;

module tb;
  mem_t m = '{4'b1111, '{8'hA5, 8'h5A}};
  initial $display("%b %h", m.opcode, m.memory.data);
endmodule
```

Q770. Is struct-to-struct assignment allowed? Give example.

Q771. Predict result:

```
typedef struct {
  int a, b;
```

```
} pair_t;

module tb;
  pair_t p1 = '{1, 2};
  pair_t p2;
  initial begin
    p2 = p1;
    $display("p2.a = %0d, p2.b = %0d", p2.a, p2.b);
  end
endmodule
```

Q772. Predict result:

```
typedef struct packed {
  bit [3:0] a;
  bit [3:0] b;
} s;

module tb;
  s s1 = '{4'b1010, 4'b0101};
  s s2 = '{4'b1010, 4'b0101};
  initial $display("Equal = %0b", s1 === s2);
endmodule
```

Q773. Describe how unions can be misused to access same memory differently. Give SystemVerilog example.

Q774. Predict value:

```
typedef union packed {
  logic [31:0] val;
  logic [3:0][7:0] bytes;
} packed_u;

module tb;
  packed_u u;
  initial begin
    u.val = 32'hAABBCCDD;
    $display("Byte[0] = %h", u.bytes[0]);
  end
endmodule
```

Q801. What are the major differences between Verilog and SystemVerilog?

Q802. What is the significance of 4-state logic in SystemVerilog?

Q803. Define ‘logic‘ in SV. How is it different from ‘reg‘ and ‘wire‘?

Q804. List all data types in SystemVerilog and categorize them.

Q805. Explain the use of ‘typedef‘ with an example.

Q806. What are packed and unpacked arrays? Provide one key application of each.

Q807. Describe the difference between ‘bit‘, ‘logic‘, and ‘reg‘.

Q808. What is meant by timeunit and timeprecision in SV?

Q809. Explain the difference between ‘assign‘ and ‘always_comb‘.

Q810. What are the different types of modeling styles in SystemVerilog?

Q811. Explain the concept of blocking and non-blocking assignments.

Q812. What are continuous assignments? Where are they used?

Q813. Describe ‘initial‘ vs ‘always‘ blocks.

Q814. What is the role of ‘generate‘ blocks in SV?

Q815. Differentiate between dynamic array, associative array, and queue.

Q816. What is ‘$size()‘, ‘$left()‘, ‘$right()‘ in SV? Provide use cases.

Q817. Explain how slicing and concatenation work on packed arrays.

Q818. What is Object-Oriented Programming in SV? How does it help verification?

Q819. Explain the concepts of ‘class‘, ‘object‘, and ‘new()‘ in SV.

Q820. What are the uses of ‘static‘, ‘const‘, and ‘virtual‘ keywords in classes?

Q821. Describe inheritance in SV with an example.

Q822. Explain ‘polymorphism‘ and its types in SystemVerilog.

Q823. What is the difference between ‘pure virtual‘ and ‘virtual‘ functions?

Q824. Define ‘this‘, ‘super‘, and ‘null‘ with suitable examples.

Q825. What is a ‘covergroup‘ and where is it used?

Q826. Define randomization in SV. What are its advantages?

Q827. What is the difference between ‘rand‘ and ‘randc‘?

Q828. How do you apply constraints to random variables?

Q829. What is ‘soft constraint‘ vs ‘hard constraint‘?

Q830. Explain inline constraints vs class-based constraints.

Q831. What is functional coverage in SV?

Q832. What are the components of a covergroup?

Q833. Define cross coverage. How is it different from individual coverage?

Q834. What are ‘bins‘, ‘illegal_bins‘, and ‘ignore_bins‘?

Q835. When would you use transition bins?

Q836. Explain ‘fork...join‘, ‘fork...join_any‘, and ‘fork...join_none‘.

Q837. What are semaphores in SV and where do we use them?

Q838. Define ‘event‘ in SV and demonstrate its use with an example.

Q839. What happens when ‘disable fork‘ is executed?

Q840. Explain mailbox usage with producer-consumer example.

Q841. What is the need for interfaces in SV?

Q842. How is a ‘modport‘ helpful inside interfaces?

Q843. Can you use tasks and functions in interfaces? If yes, how?

Q844. What is the difference between task and function?

Q845. Can tasks return values in SV? Explain.

Q846. Explain scope rules inside tasks and functions.

Q847. What are automatic vs static functions/tasks?

Q848. What is the role of ‘void‘ function in SV?

Q849. What is a handle in SystemVerilog? How is memory managed?

Q850. Can you instantiate a class without using ‘new()‘?

Q851. Explain shallow copy vs deep copy with code.

Q852. What is a factory in SV? How does it help in reuse?

Q853. Can constraints be written outside the class? Justify.

Q854. What is distribution constraint ‘‘ used for?

Q855. Define solve-before constraint in detail.

Q856. How do you randomize arrays with unique values?

Q857. What are assertions in SystemVerilog?

Q858. Differentiate between immediate and concurrent assertions.

Q859. Write an assertion to check that ‘a‘ becomes high 2 cycles after ‘b‘.

Q860. What is implication operator in property assertions?

Q861. Explain the use of ‘$display‘, ‘$monitor‘, and ‘$strobe‘.

Q862. What is the role of ‘$random‘, ‘$urandom‘, and ‘$urandom_range()‘?

Q863. How is ‘$fatal‘ different from ‘$error‘?

Q864. Compare struct vs union in SV.

Q865. What is the difference between packed and unpacked struct?

Q866. Can an enum be used in a struct? Justify with an example.

Q867. Can unions have arrays? Provide a valid example.

Q868. What is DPI in SV? Where is it used?

Q869. How do you import a C function in SystemVerilog?

Q870. What are the restrictions when calling DPI functions?

Q871. How are SVA used to capture corner case bugs?

Q872. Write a temporal assertion for detecting illegal burst.

Q873. Can you override class constraints dynamically?

Q874. What is the difference between coverage-driven and directed testing?

Q875. Explain ‘typedef enum logic‘ vs ‘typedef enum bit‘.

Q876. Which SystemVerilog features are synthesizable?

Q877. Why are classes not synthesizable?

Q878. Explain which types of arrays are synthesis-friendly.

Q879. What is the role of ‘interface‘ in synthesizable design?

Q880. Create a 4x1 MUX using ‘assign‘, ‘always_comb‘, and ‘case‘.

Q881. Design a memory controller using struct and enum.

Q882. Use covergroup to check FIFO occupancy bins.

Q883. Use constraint to generate a unique 8-bit sequence.

Q884. Explain race condition and ways to avoid it.

Q885. What is the role of ‘unique‘ and ‘priority‘ in ‘case‘ statements?

Q886. Can an interface be passed into a class? How?

Q887. How is ‘randcase‘ different from ‘case‘?

Q888. What does ‘$bits()‘ return when applied to a packed struct?

Q889. Explain ‘$isunknown()‘ and when it is useful.

Q890. What happens if we randomize an object with ‘null‘ fields?

Q891. Why is functional coverage more important than code coverage?

Q892. How do you achieve reusability in testbenches?

Q893. What are the common SystemVerilog features used in UVM?

Q894. Explain the difference between passive and active agents in testbench.

Q895. Why is fork-join_none useful in verification environments?

Q896. List 10 commonly used SV system tasks and their purpose.

Q897. What are the different return types allowed in functions?

Q898. Can you use ‘$cast‘ for type conversion in SV?

Q899. How does ‘ref‘, ‘const ref‘, and ‘input‘ behave differently in tasks?

Q900. Which SystemVerilog feature do you personally find most powerful and why?

Q901. What is a mailbox in SystemVerilog? Describe its purpose.

Q902. Differentiate between ‘mailbox‘, ‘semaphore‘, and ‘event‘.

Q903. What is the default size of a mailbox?

Q904. Write syntax to declare and create a mailbox object.

Q905. How is ‘mailbox.put()‘ different from ‘mailbox.try_put()‘?

Q906. How do ‘put()‘ and ‘get()‘ methods behave in blocking vs non-blocking mode?

Q907. How would you create a bounded mailbox of size 5?

Q908. What happens when a process tries to ‘get()‘ from an empty mailbox?

Q909. Is it possible to peek into a mailbox without removing the item? How?

Q910. Explain the role of ‘peek()‘ in producer-consumer environments.

Q911. What is a semaphore in SystemVerilog and what problem does it solve?

Q912. What is the syntax for creating and initializing a semaphore with 3 keys?

Q913. Explain the difference between ‘get()‘ and ‘try_get()‘ in semaphores.

Q914. How does ‘put()‘ work in semaphore? Give an example.

Q915. What is the use of semaphore in a critical section of code?

Q916. Can you release more keys than you acquired using ‘put()‘? What will happen?

Q917. Demonstrate a scenario where semaphore prevents race conditions.

Q918. Can semaphores be used for signaling? Justify your answer.

Q919. Write a simple producer-consumer example using mailbox.

Q920. Create a scenario where two consumers wait for one producer via mailbox.

Q921. Build a simple testbench to show message passing between modules using mailbox.

Q922. Implement thread synchronization using semaphore between two parallel threads.

Q923. Use semaphore to limit access to a shared resource for only 2 out of 5 threads.

Q924. Use mailbox to transfer class-based transactions between driver and monitor.

Q925. Why is peek() rarely used compared to get() in real environments?

Q926. What will happen if mailbox is full and you call ‘put()‘? Will it block?

Q927. How do you debug a deadlock caused by mailbox blocking calls?

Q928. Can you pass objects through a mailbox? What precautions should you take?

Q929. When using mailbox, how would you terminate gracefully when the mailbox is empty?

Q930. Is semaphore thread-safe? Justify your answer.

Q931. Compare and contrast ‘event‘, ‘mailbox‘, and ‘semaphore‘ with real-life examples.

Q932. Which is more suitable for a producer-consumer architecture: mailbox or semaphore?

Q933. Can mailbox be used to signal multiple waiting threads?

Q934. Why are mailboxes better for transferring data than events?

Q935. What is the advantage of using semaphore in scoreboards?

Q936. Build a transaction-level model where mailboxes are used for inter-module communication.

Q937. Use a mailbox in conjunction with a ‘fork...join‘ to create multi-threaded data flow.

Q938. Implement a mailbox that handles class transactions with sequence numbers.

Q939. Write a function that uses a semaphore to control access to a shared log file.

Q940. Can you share a mailbox between multiple modules? How would you do it?

Q941. How would you detect a starvation issue with mailbox?

Q942. What is a potential drawback of using a blocking ‘get()‘ in mailbox?

Q943. Can semaphores lead to deadlocks? How can this be avoided?

Q944. How do semaphores help when creating critical sections in concurrent environments?

Q945. How are mailboxes implemented internally (conceptual understanding)?

Q946. Can mailbox be passed through virtual interfaces or classes? Explain.

Q947. What is the purpose of ‘try_get()‘ in semaphore or mailbox and when is it preferred?

Q948. How do you perform multi-thread transaction-level communication using mailbox?

Q949. Explain a use case where mailbox + semaphore are used together.

Q950. Why is mailbox an effective synchronization mechanism in a UVM environment?

Q951. Predict the output:

```
class A;
  function void show(); $display("A"); endfunction
endclass

class B extends A;
  function void show(); $display("B"); endfunction
endclass

A obj = new;
obj = new B();
obj.show();
```

Q952. Predict the output when a virtual function is overridden and called via base class handle.

Q953. What happens if the base class does not declare the function as virtual but it is overridden in the child?

Q954. Consider multiple classes extending each other. How is ‘super‘ invoked in chain?

Q955. Write output for:

```
class X;
  static int val = 10;
  function void print(); $display("val=%0d", val); endfunction
endclass

X x1 = new(); x1.val = 20;
X x2 = new(); x2.print();
```

Q956. Predict the output:

```
initial begin
  fork
    #5 $display("A");
    #10 $display("B");
  join
  $display("Done");
end
```

Q957. What will be printed and in what order if you use ‘join_any‘?

Q958. Replace ‘join‘ with ‘join_none‘. Will ”Done” print immediately?

Q959. Predict the output with nested fork-join and delays.

Q960. Modify a thread to have a ‘disable fork‘. What impact will it have?

Q961. Predict:

```
semaphore s = new(1);
initial begin
  fork
    begin s.get(1); #5; $display("T1"); s.put(1); end
    begin s.get(1); $display("T2"); s.put(1); end
  join
end
```

Q962. Use ‘try_get()‘ instead of ‘get()‘. Will both threads succeed?

Q963. Predict order of execution if you increase semaphore size to 2.

Q964. How many threads can access a shared section with ‘semaphore s = new(3);‘?

Q965. If thread forgets to ‘put()‘, what happens?

Q966. Predict the output:

```
mailbox m = new();
initial begin
  fork
    begin #5 m.put(100); end
    begin int x; m.get(x); $display("Received: %0d", x); end
  join
end
```

Q967. What if ‘put()‘ is delayed longer than ‘get()‘?

Q968. Will ‘peek()‘ remove element from mailbox?

Q969. Use ‘try_get()‘ on empty mailbox — what will be the output?

Q970. Demonstrate deadlock condition using mailbox.

Q971. Predict:

```
int arr[3] = '{1,2,3};
foreach (arr[i]) begin
  arr[i] = arr[i] * 2;
end
$display("%p", arr);
```

Q972. What is the output of dynamic array initialized with ‘new[5]‘ but only partially filled?

Q973. Queue manipulation:

```
int q[$] = {1,2,3};
q.push_back(4); q.pop_front();
$display("%p", q);
```

Q974. Predict what happens when ‘associative_array[”key”]‘ is accessed but not initialized?

Q975. Use ‘foreach‘ on associative array. What is order?

Q976. Create a class with mailbox as a data member. Use it to transfer between threads. Predict result.

Q977. Use fork-join with mailbox: one thread puts, another gets. Who blocks?

Q978. Mix semaphore and mailbox – who gets control first?

Q979. Can static variable across multiple class objects retain value? Predict behavior.

Q980. Class inheritance with array of objects. Modify in base handle, read in child – what happens?

Q981. Use mailbox with multiple put/get and predict final size of mailbox.

Q982. Create 2 classes, both use same semaphore – predict interleaving of access.

Q983. Can nested classes access parent's static variable?

Q984. Predict result of 'try_get()' in fork-join_none if mailbox is empty at start.

Q985. Create fork block with delay, put into mailbox inside, get outside — when will 'get()' complete?

Q986. What is output if 'put()' puts a class object and 'get()' modifies it? Shared or copied?

Q987. Predict what happens with 'semaphore.get(2); semaphore.put(1);' and another thread tries 'get(1);'

Q988. Predict effect of this sequence: 'fork-join_none' + 'mailbox.get()' + 'disable fork;'

Q989. Does order of 'put()' and 'get()' affect result in race condition? Explain via example.

Q990. Predict mailbox behavior when class objects are passed by handle – any side effects?

Q991. Combine fork, class, and mailbox – will message reach other thread?

Q992. Predict output of overriding virtual method inside fork block.

Q993. Semaphore used to serialize access, but someone forgets 'put()' – what will deadlock look like?

Q994. Predict:

```
mailbox m = new(); int x;
m.put(5); m.peek(x); m.get(x);
$display("x=%0d", x);
```

Q995. Queue inside class is randomized — predict elements after 3 iterations.

Q996. Class with static int shared across threads – predict value after multiple increments.

Q997. Dynamic array resized in one thread – is it visible to another thread?

Q998. 'semaphore.get()' blocks, then 'fork...join_none' terminates — is 'get()' completed?

Q999. Mailbox shared across multiple class objects – predict conflict scenarios.

Q1000. Predict if below code deadlocks:

```
semaphore s = new(1);
initial begin
  fork
    begin s.get(); #10; s.put(); end
    begin #5; s.get(); s.put(); end
  join
end
```