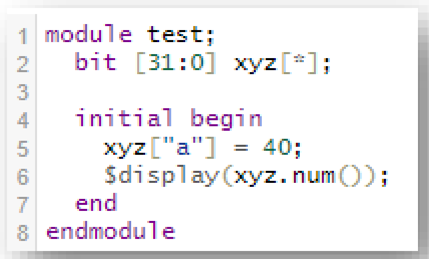
Ain Shams University
Faculty of Engineering
Computer and Systems Department
Spring 2023

CSE 313s
Selected Topics in Computer Engineering
Sheet 4

1. Which of the following are true with respect to System Verilog arrays?
   a. Associative arrays can be used when size of an array is not known as it can be built as key/value pairs.
   b. Dynamic arrays are useful for contiguous collection of variables whose number keeps varying.
   c. Dynamic arrays can be re-sized after size is allocated.
   d. [✓ All of the above]

2. What will be the output of the code below?



```
module test;
  bit [31:0] xyz[*];

  initial begin
    xyz["a"] = 40;
    $display(xyz.num());
  end
endmodule
```

   a. a [✓ b. 1] c. 40 d. 41

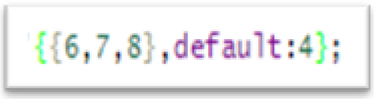
3. Logic variables can have multiple drivers.
   a. True
   [✓ b. False]

4. ‘bit’ is two state
   a. [✓ True]
   b. False

5. Testbench functionality is to:
   a. Generate stimulus and apply to DUT
   b. Capture response and check for correctness
   c. Measure progress against verification goals
   d. [✓ All of the above]

6. ‘wire’ variables can have multiple drivers
   a. [✓ True]
   b. False

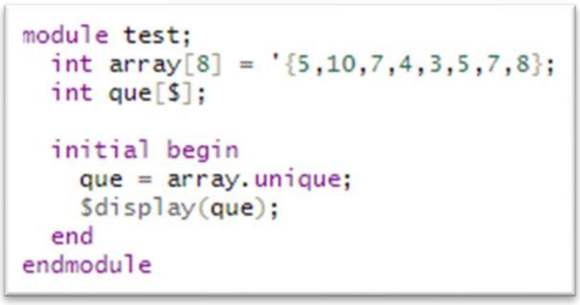
7. The declaration of the array shown in the following code is:



```
'{{6,7,8},default:4};
```

   a. array_m[3][2]
   b. array_m[1][2]
   c. [✓ array_m[2][3]]
   d. array_m[3][3]

8. What will be the output of the below code:



```
module test;
  int array[8] = '{5,10,7,4,3,5,7,8};
  int que[$];

  initial begin
    que = array.unique;
    $display(que);
  end
endmodule
```

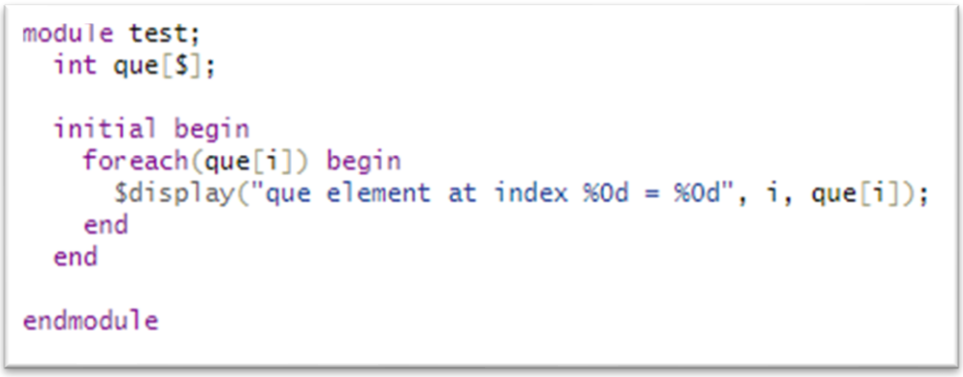
   a. Compilation error
   b. [✓ ‘{{5, 10, 7, 4, 3, 8}]
   c. ‘{3, 4, 5, 5, 7, 7, 8, 10}
   d. ‘{5, 10, 7, 4, 3, 5, 7, 8}

9. With respect to dynamic array, which of the following are true (select all those apply)
   a. [✓ During runtime, array can be extended and retaining the old values]
   b. Can add and remove array elements from anywhere of the array
   c. [✓ Array can grow and shrink at runtime]
   d. [✓ Array is initially empty and space is allocated when new[ ] is called]

10. Array [4][8]. This array declaration type is:
   a. [✓ Compact declaration]
   b. Verbose declaration
   c. Single dimension array declaration
   d. None of the above

11. Which of the following is queue declaration?
   a. int a[ ];
   b. [✓ int a[$];]
   c. int a;
   d. int a[*];

12. What will be the output of the below code?



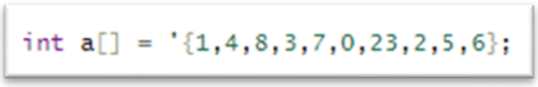
```
module test;
  int que[$];

  initial begin
    foreach(que[i]) begin
      $display("que element at index %0d = %0d", i, que[i]);
    end
  end

endmodule
```

a. Fatal error
b. Que element at index 0 = 0
c. [✓ Simulation ends without displaying display statement]
d. Compilation error

13. Does SystemVerilog support multidimensional arrays?
a. [✓ True]
b. False

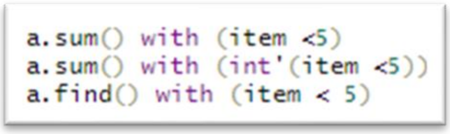
14. Is there any method to append one dynamic array to another? z={x,y}

15. Given a is an array defined as follows:



```
int a[] = '{1,4,8,3,7,0,23,2,5,6};
```

What is the difference between the following statements?



```
a.sum() with (item <5)
a.sum() with (int'(item <5))
a.find() with (item < 5)
```

1
5
{1,4,3,0,2}

16. What is the difference between Associative array and Dynamic array?

17. What's the difference between data type logic, reg and wire?

18. How many array types in SystemVerilog? How do you use them? Show examples.

19. What is the difference between a bit and logic data type?

20. What is the difference between logic[7:0] and byte variable in SystemVerilog?

21. Which of the array types: dynamic array or associative array, are good to model really large arrays, say: a huge memory array of 32KB?

22. Suppose a dynamic array of integers (myvalues) is initialized to values as shown below. Write a code to find all elements greater than 3 in the array using array locator method *find*?

    int myvalues[ ] = '{9, 1, 8, 3, 2, 4, 6};

23. Given a dynamic array of size 20, how can the array be re-sized to hold 40 elements while the lower 20 elements are preserved as original?

Assignment
Due date: April 3rd, 2023

Given the following specifications, write a SV code to describe the system.

You have a multi-mode counter. It can count up, down, by ones and by twos. There is a two-bit control bus input indicating which one of the four modes is active.

Control Value	Function
00	Count up by 1's
01	Count up by 2's
10	Count down by 1's
11	Count down by 2's

You also have an initial value input and a control signal called INIT. When INIT is 1, the initial value is parallelly loaded into the counter.

Whenever the count is equal to all zeros, set a signal called LOSER high. When the count is all ones, set a signal called WINNER high. In either case, the set signal should remain high for only one cycle.

With a pair of plain binary counters, count the number of times WINNER and LOSER goes high. When one of them reaches 15, set an output called GAMEOVER high. If the game is over because LOSER got to 15 first, set a two-bit output called WHO to 2'b01. If the game is over because WINNER got to 15 first, set WHO to 2'b10. WHO should start at 2'b00 and return to it after each game over.

Then synchronously clear all the counters and start over. The game never ends.

Does your design need a clock? Does it need an asynchronous reset? A synchronous reset? Decide for yourself and explain your choices.

You are not limited in either the number of functional blocks or the number of modules used.

Develop a simple test bench that generates the required inputs to test the different scenarios, and make sure the design is correct by investigating the generated wave diagrams.