Q1)
   a) Is there a case for randomly moving the DUT into erroneous states? Explain.

   b) When randomizing the input values into a DUT, would you attempt to
      randomize over the entire space of values it would take? For example, if you had a
      1K byte addressable memory (1 address corresponds to 1 byte) and a 16 bit address
      value, how would your randomization approach be?
Q2) As a verification engineer, you are yet to reach 100% coverage. You have a coverage graph that shows the trend below. What would your course of action be? What aspects of the design would you consider when making this decision?
Q3) Consider the following piece of code in the program body. Please provide the results ($display statements).

```plaintext
int q[] = {0, 1, 2, 3, 4, 5, 12, 2, 4, 8};
int b[] = '{2{3}};
int c[];
int i,j;

b.push_front(6);
c = q;
c.push_back(b[0]);
c.push_front(b[2]);

foreach(c[i])
  $display("%d", c[i]);

j = b.pop_back();
c.insert(6,b[1]);
c.insert(6,b[-1]);
c.insert(c.size()-2, j);

foreach(c[i]) // display the contents of c
  $display("%d", c[i]);

if(c.size() !=0)
  c = {};

c = q.find_index with(item <3);
foreach(c[i])
  b.push_front(q[c[i]]+1);

foreach(b[i]) // display the contents of b
  $display("%d", b[i]);
```
Q4) Consider the following piece of code in the program body. Please provide the results ($display statements).

// declaration
bit [1:0][1:0][31:0]    temp [2];

temp[0] = 128'h1234_bace_face_dead_0fee_5678_9abc_defa;
temp[1][0] = 64'hbead_1001_badd_d00d;
temp[1][1][1] = 32'hcadd_0cea;

$display("%h", temp[0][1]);

$display("%h", temp[0][0][0][1]);

$display("%h", temp[1][0][1]);
Q5) Consider the memory arrangement shown below:

Let us assume that the memory array is called “Mem”. Also the memory is readable at the a, b, c and d dimensions.

a) write the declaration of this memory

b) Use ONE foreach loop to display the contents of the entire memory along the x and y dimensions.

c) Use nested for loops to display each content at the level of a, b, c and d under dimension x = 4 and dimension y = 12 of the memory.
Q6) Consider the following program body

```markdown
bit [2:0] array[2] [6] = '{2, 2, 0, 1, 1, 1}, '{2, 2, 2, 0, 2, 4});
int result1, result2;
result1 = array[0].sum;
result2 = array[1].sum;
$display("array[0].sum = %d; result1 = %d; array[1].sum = %d; result2 = %d; ", array[0].sum, result1, array[1].sum, result2);
```

Provide the result of this program. Explain yourself.
Q7) Please code up the missing parts and provide the result of the program.

```
byte a[];
byte b[];

initial
begin

    // allocate 7 elements for a[]

    // initialize the array to {a, 3, 2, 7, f, 4}

    // copy a to b and expand the number of entries to 12: ONE COMMAND
    // PLEASE

    foreach(b[i])
        $display(b[i])

end
```
Q8) Please code in the sections missing.

bit[0:23]  a[*];
bit[0:23]  b[string];
bit[0:23]  val;

val = 12;
initial
begin

repeat (5)
begin
index = $random(0, 65534);
a[index] = val;
val = val + 10;
end

// exchange the 1st and the last elements of the associative array “a”

// display the contents of the associative array: use foreach please

// remove the 3rd element in the associative array.

// create the associative array “b” that contains the following table
// of values in order
//    122 Fahrenheit
//    50  Celsius
//    323 Kelvin

// write a display statement that would show the value of the Celsius
// index

// modify the Kelvin entry to 423 in b

End
Q9)

a) Define a data-type called "packet" with the following memory arrangement:

```
<table>
<thead>
<tr>
<th>seq_num</th>
<th>src</th>
<th>dst</th>
<th>payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>8 bit signed</td>
<td>8 bit signed</td>
<td>1024 bit unsigned value</td>
</tr>
</tbody>
</table>
```

// Declare an fixed array of size 128 of the type shown above called pkt_array

// Modify the src field of pkt_array at index 10 to 64 .. ONE COMMAND ONLY PLEASE

// change the 24th and 25th bits of pkt_array at index 2 to 0 and 1 .. ONE COMMAND ONLY PLEASE

b) How many bits does the final structure consume?
Q10)

// a) Create an enumerated type called state with the following
// assignment of names to values where the values are of type 3
// bit sized: GET = 2; SEND = 3; WAIT = 5; IDLE = 7

// b) Declare 2 variables fsm1 and fsm2 of type state

// Display the value and the associated sting value of the third
// member of the enumeration