Contents

SystemVerilog

Name: Shefin Muhammed MS

Topic: Practice Codes

1. 30-10-2025

1.1. Fixed array

Code:

```
class fixed_array;
int arr[10]; //fixed array

function new();
foreach(arr[i])begin
arr[i]=i*i;
end
endfunction

function void display();
foreach(arr[i])begin
$display("Fixed array value Arr[%0d],= %0d",i,arr[i]);
end

endfunction
endclass

module fixed_array_with_class;

fixed_array arry;

initial begin
arry=new();
arry.display();
end

endmodule
```

Output:

```
# Fixed array value Arr[0],= 0
# Fixed array value Arr[1],= 1
# Fixed array value Arr[2],= 4
# Fixed array value Arr[3],= 9
# Fixed array value Arr[4],= 16
# Fixed array value Arr[5],= 25
# Fixed array value Arr[6],= 36
# Fixed array value Arr[7],= 49
# Fixed array value Arr[8],= 64
# Fixed array value Arr[9],= 81
```

2. 03-11-2025

2.1. Simple Example code for Shallow Copy

Code:

```
class example;
  string name;
endclass

module test;
  example n1,n2;

  initial begin
    n1=new();
    n1.name="Shefin";

    n2=n1;
    $display("
        ----------------------------------------------------------------")
        ;
    $display("Before change: n1.name = %s, n2.name = %s", n1.name, n2.
        name);
    $display("
        ----------------------------------------------------------------")
        ;

    n2.name="Gayu";
    $display("after change: n1.name = %s, n2.name = %s", n1.name, n2.
        name);

  end

endmodule
```

Output:

```
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL: ----------------------------------------------------------------
# KERNEL: Before change: n1.name = Shefin, n2.name = Shefin
# KERNEL: ----------------------------------------------------------------
# KERNEL: after change: n1.name = Gayu, n2.name = Gayu
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
```

2.2. Example for Deep copy

Code:

```
class deepcpy;
  string name;

  function deepcpy copy();
    deepcpy temp=new();
    temp.name=this.name;
    return temp;

  endfunction

endclass

module test;
  deepcpy n1,n2;

  initial begin
    n1=new();
    n1.name="Shefin";

    n2=n1.copy();

    $display("
       ----------------------------------------------------------------")
       ;
    $display("Before change: n1.name = %s, n2.name = %s", n1.name, n2.
       name);
    $display("
       ----------------------------------------------------------------")
       ;

    n2.name="Gayu";
    $display("after change: n1.name = %s, n2.name = %s", n1.name, n2.
       name);

    $display("--------------------------E-N-D
       -------------------------------");


  end

endmodule
```

```
# KERNEL: ----------------------------------------------------------------
# KERNEL: Before change: n1.name = Shefin, n2.name = Shefin
# KERNEL: ----------------------------------------------------------------
# KERNEL: after change: n1.name = Shefin, n2.name = Gayu
# KERNEL: -------------------------E-N-D--------------------------------
```

3. 04-11-2025

3.1. Inheritance

code:

```
    // example code for inheritance
class parent_trans;//parent class
  bit[31:0]data;
  int id;

  function void display();
    $display("The value are data=%0d||id=%0d",data,id);
  endfunction

endclass

class child_1 extends parent_trans; //acessing parent class using
    extends keyword
  bit[31:0]data;

  function void display1();
    $display("The value are data=%0d||id=%0d",data,id);
  endfunction
endclass

module inheeritance_example;
  child_1 ch1;//class handle ch1
  initial begin
    $display("----------------------------Start
       -------------------------");
    ch1=new();
    ch1.data=200;//data value update to 200
    ch1.id=5;//id value in parent update to 5 because parent is
       connected to child using extends

    ch1.display1();//display ch1 display || data =200, id=5

    ch1.display();//display parent display || data =0, id=5
    $display("----------------------------END-------------------------"
       );
  end

endmodule
```

Output

```
# KERNEL: ----------------------------Start-------------------------
# KERNEL: The value are data=200||id=5
# KERNEL: The value are data=0||id=5
# KERNEL: ----------------------------END-------------------------
```

3.2. inheritance example with multiple child

code:

```
class parent_trans;
  bit[31:0]data_p;
  int id_p;
endclass

class child1 extends parent_trans;
  bit[31:0] data_ch1;
   int id_ch1;
endclass

class child2 extends parent_trans;
  bit[31:0]data_ch2;
  int id_ch2;
endclass

class child1_1 extends child1;
  bit[31:0]data_ch1_1;
   int id_ch1_1;
endclass

class child1_2 extends child1_1;
  bit[31:0]data_ch1_2;
   int id_ch1_2;
endclass

class child1_3 extends child1_2;
  bit[31:0]data_ch1_3;
  int id_ch1_3;
endclass


module example_inheritance;
  child1_3 ch1_3;
  child2 ch2;
  initial begin
    ch1_3=new();
    ch2=new();
    $display("--------------------start--------------------");
    ch1_3.id_ch1_3=20;
    ch1_3.id_p=10;
    ch1_3.data_p=100;

    ch1_3.data_ch1_2=600;

    ch2.data_ch2=90;

    $display("The value of id and data are [data_ch1_2=%0d],[data_p=%0d
       ],[id_p=%0d],[id_ch1_3=%0d] ",ch1_3.data_ch1_2, ch1_3.data_p ,
       ch1_3.id_p ,ch1_3.id_ch1_3);

    $display("--------------------END--------------------");
  end

endmodule
```

output

```
# KERNEL: --------------------start-------------------
# KERNEL: The value of id and data are [data_ch1_2=600],

[data_p=100],[id_p=10],[id_ch1_3=20]
# KERNEL: The child 2 value is data_ch2=90
# KERNEL: ---------------------END--------------------
```

3.3. Over riding

code:

```
    //over riding example
class parent_trans;//parent class
  bit[31:0]data=100;
  int id=5;

  function void display1();//For over riding the data we need same
     function name and same no of arguments
    $display("The value are data=%0d||id=%0d",data,id);
  endfunction

endclass

class child_1 extends parent_trans; //acessing parent class using
   extends keyword
  bit[31:0]data=200;
  int id = 10;

  function void display();
    $display("The value are data=%0d||id=%0d",data,id);
  endfunction
endclass

module example;
  child_1 ch1;//class handle ch1
  initial begin
    $display("----------------------------Start
       -------------------------");
    ch1=new();
    ch1.display();//display child value data=200,id=10;
    //because by using the same function name and argument which
       overwrite the value of the first display
    //in first the value of data is 100 and id = 5,we can see those
       value if we change the display name. for ex:display2
    ch1.display1();
    $display("----------------------------END-------------------------"
       );
  end

endmodule
```

Output

```
KERNEL: The value are data=200||id=10
```

3.4. super keyword

Code:

```
//super keyword
class parent_trans;//parent class
  bit[31:0]data;
  int id;

  function void display_p();
    $display("base:The value are data=%0d||id=%0d",data,id);
  endfunction
endclass

class child_1 extends parent_trans; //acessing parent class using
   extends keyword
  bit[31:0]data;

  function void display();
    super.data=3;//super keyword can access the parent data inside
       child class
    super.id=40;
    $display("child:The value are data=%0d",data);
  endfunction
endclass

module example;
  child_1 ch1;//class handle ch1
  initial begin
    $display("----------------------------Start
       -------------------------");
    ch1=new();
    ch1.data=20;
    ch1.display();
    ch1.display_p();//parent value is upadted using Super keyword
    $display("----------------------------END-------------------------"
       );
  end

endmodule
```

Output

```
# KERNEL: child:The value are data=20
# KERNEL: base:The value are data=3||id=40
```

4. Practice questions

4.1. Question 1

- IN ARRAY ODD LOCATION STORED WITH EVEN DATA ,EVEN LOCATION STORED WITH EVEN DATA?

Code:

```
class question1;

  rand int arr[20];

  constraint ans {
    foreach(arr[i]) {
      if (i % 2 == 0) // checking for even location
        (arr[i] % 2 == 1)&&(arr[i] inside {[0:100]}); // odd data in
           even location

      else           // odd location
        (arr[i] % 2 == 0)&&(arr[i] inside {[0:100]});// even data in
           odd location
    }
  }

endclass


module test;

  question1 q1;

  initial begin
    q1=new();

    q1.randomize();

    $display("array values are:");
    foreach(q1.arr[i])
      $display("array[%0d] = %0d",i,q1.arr[i]);
  end


endmodule
```

output:

```
# KERNEL: array values are:
# KERNEL: array[0] = 7
# KERNEL: array[1] = 18
# KERNEL: array[2] = 39
# KERNEL: array[3] = 78
# KERNEL: array[4] = 61
# KERNEL: array[5] = 36
# KERNEL: array[6] = 61
# KERNEL: array[7] = 66
```

```
# KERNEL: array[8] = 37
# KERNEL: array[9] = 8
# KERNEL: array[10] = 89
# KERNEL: array[11] = 62
# KERNEL: array[12] = 89
# KERNEL: array[13] = 30
# KERNEL: array[14] = 47
# KERNEL: array[15] = 4
# KERNEL: array[16] = 5
# KERNEL: array[17] = 58
# KERNEL: array[18] = 23
# KERNEL: array[19] = 84
```

4.2. Question 2

- Write a SystemVerilog program with two tasks running in parallel.
- Task-1 should generate numbers from 0 to a maximum value and trigger an event whenever the number is even.
- Task-2 should wait for this event and print the even number along with simulation time.
- Use fork–join to run both tasks concurrently.

```
module two_task_question;
int even;//for saving the even i value
int i,j;//for loop
event t1;//calling event t1
int max=50;//changing the value

  task task1();//task 1 for checking the i value is even or not
    for (i=0;i<=max;i=i+1)begin
     if(i%2==0)begin
       even=i;
       #1//using 1ns delay for task2;
      ->t1;//triggering event t1;
     end
    end
  endtask

  task task2();//to print the even value using task 2
    for(j=0;j<max/2;j=j+1)begin
      @(t1);
      $display("@%0t :Task2 : Even value detected = %0d",$time,even);
     end
  endtask

  initial begin
    fork
      task1();
      task2();
    join
  end
endmodule
```

Output

```
# KERNEL: @1 :Task2 : Even value detected = 2
# KERNEL: @2 :Task2 : Even value detected = 4
# KERNEL: @3 :Task2 : Even value detected = 6
# KERNEL: @4 :Task2 : Even value detected = 8
# KERNEL: @5 :Task2 : Even value detected = 10
# KERNEL: @6 :Task2 : Even value detected = 12
# KERNEL: @7 :Task2 : Even value detected = 14
# KERNEL: @8 :Task2 : Even value detected = 16
# KERNEL: @9 :Task2 : Even value detected = 18
# KERNEL: @10 :Task2 : Even value detected = 20
```

```
# KERNEL: @11 :Task2 : Even value detected = 22
# KERNEL: @12 :Task2 : Even value detected = 24
# KERNEL: @13 :Task2 : Even value detected = 26
# KERNEL: @14 :Task2 : Even value detected = 28
# KERNEL: @15 :Task2 : Even value detected = 30
# KERNEL: @16 :Task2 : Even value detected = 32
# KERNEL: @17 :Task2 : Even value detected = 34
# KERNEL: @18 :Task2 : Even value detected = 36
# KERNEL: @19 :Task2 : Even value detected = 38
# KERNEL: @20 :Task2 : Even value detected = 40
# KERNEL: @21 :Task2 : Even value detected = 42
# KERNEL: @22 :Task2 : Even value detected = 44
# KERNEL: @23 :Task2 : Even value detected = 46
# KERNEL: @24 :Task2 : Even value detected = 48
# KERNEL: @25 :Task2 : Even value detected = 50
```

4.3. Question 3

- take one Enum inside emplementt two or three like low medium high ,
- if Enum is low take different 3 arrays one array take the value below 100
- must be devisible by 5,if Enum is medium any array must be generate prime numbers,
- Enum is high then genretae form 0 to 100

```
  class question3;
int arr1[];
int arr2[];
int arr3[];

typedef enum {LOW, MID, HIGH} state;
state s1;

function void value(state st);
    s1 = st;
endfunction

function int prime(int num);
  if(num <= 1)
    return 0;

  for(int i = 2; i*i <= num; i++) begin
    if(num % i == 0)
      return 0;
  end

  return 1;
endfunction


// ---------------- LOW ----------------
function void low();
```

```
    int count = 0;
    int index = 0;

    if(s1 == LOW) begin

      // count first (correct)
      for(int i = 0; i < 100; i++) begin
        if(i % 5 == 0)
          count++;
      end

      // allocate array AFTER counting
      arr1 = new[count];

      // store values
      for(int i = 0; i < 100; i++) begin
        if(i % 5 == 0) begin
          arr1[index] = i;
          index++;
        end
      end

    end
  endfunction


  // ---------------- MID ----------------
  function void mid();
    if(s1 == MID) begin
      int count = 0;
      int index = 0;

      // count primes
      for(int i = 0; i <= 100; i++)
        if(prime(i))
          count++;

      arr2 = new[count];


      for(int i = 0; i <= 100; i++) begin
        if(prime(i)) begin
          arr2[index] = i;
          index++;
        end
      end
    end
  endfunction


  // ---------------- HIGH ----------------
  function void high();
    if(s1 == HIGH) begin
      arr3 = new[101];
      for(int i = 0; i <= 100; i++)
        arr3[i] = i;
    end
  endfunction
```

```
86
87  endclass
88
89
90  // ================= TEST MODULE =================
91
92  module test_q3;
93    question3 q3;
94
95    initial begin
96      q3 = new();
97
98      q3.value(question3::LOW);
99      q3.low();
100     $display("LOW arr1 = %p", q3.arr1);
101
102     q3.value(question3::MID);
103     q3.mid();
104     $display("MID arr2 = %p", q3.arr2);
105
106     q3.value(question3::HIGH);
107     q3.high();
108     $display("HIGH arr3 = %p", q3.arr3);
109   end
110
111 endmodule
```

output

```
# LOW arr1 = '{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
60, 65, 70, 75, 80, 85, 90, 95}
# MID arr2 = '{2, 3, 5, 7, 11, 13, 17, 19, 23, 29,31,37,41, 43, 47, 53, 59, 61,67,
71, 73, 79, 83, 89, 97}
# HIGH arr3 = '{0, 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, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64,
65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,77, 78, 79, 80, 81,
82,83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99, 100}
```

4.4. Question 4

- generate the array value without using unique keyword and rand c within the range of 50 to 100;

```
\\method1
class question_4;
  rand int arr[10];

  constraint range {
    foreach(arr[i])
      arr[i] inside {[50:100]};
  }

  function void post_randomize();
    foreach(arr[i])begin
      foreach(arr[j])begin
        if(arr[i]==arr[j]&&i!=j)begin
          arr[j]=arr[j]+1;
        end
      end
    end
  endfunction
endclass

module test;
  question_4 q3;
  initial begin
    q3=new();
    repeat(5)begin
      q3.randomize();
      $display("arr= %p",q3.arr);
    end
  end
endmodule

\\method 2

class prim_array;
  rand int arr[10];
  constraint range_c {
    foreach (arr[i]) arr[i] inside {[0:50]};
  }

  constraint pri {
    foreach (arr[i])
      foreach (arr[j])
        if (i < j)
          arr[i] != arr[j];
  }

  function void print();
    $display("array=%p ", arr);
  endfunction
endclass

module tb;
  prim_array pa;
```

```
54
55 initial begin
56   pa = new();
57   repeat(5)begin
58   pa.randomize();
59   pa.print();
60   end
61 end
62 endmodule
```

output

```
# KERNEL: arr= '{53, 86, 52, 99, 95, 90, 77, 50, 87, 62}
# KERNEL: arr= '{88, 58, 72, 95, 64, 67, 62, 80, 77, 100}
# KERNEL: arr= '{84, 68, 75, 73, 58, 76, 82, 74, 83, 79}
# KERNEL: arr= '{76, 69, 93, 88, 78, 71, 72, 100, 65, 50}
# KERNEL: arr= '{93, 65, 91, 69, 70, 92, 50, 95, 79, 84}
```

4.5. Question 5

- Write a constraint to generate a random number between 1 and 100.

```
1  class question1;
2  rand int a;
3
4    constraint value{a inside {[1:100]};}//constraint for selecting
       number between 1 to 100
5
6  endclass
7
8  module answer;
9    question1 q1;
10
11   initial begin
12     q1=new();
13     repeat(5)begin
14       q1.randomize();
15       $display("The Random generated value is %0d",q1.a);
16     end
17   end
18
19 endmodule
```

Output:

```
# KERNEL: The Random generated value is 32
# KERNEL: The Random generated value is 54
# KERNEL: The Random generated value is 73
# KERNEL: The Random generated value is 40
# KERNEL: The Random generated value is 15
```

4.6. Question 6

- Generate an even number between 10 and 50.

```
class question2;
rand int a;//rand keyword for generating random value

  constraint value {a inside {[10:50]};
                    (a%2==0);} //constraint for value from 50 to 100 ,
                       using modules to check the reminder is 0 or 1
                       for even number

endclass

module answer2;
  question2 q2;//class handle

  initial begin
    q2=new();//memory allocation for object
    repeat(5)begin

      if(!q2.randomize())//checking randomize return 0 or 1 ; if 1
         success,0 false;
        $display("Constraint Failed");
      else
      $display("generated even number is  %0d",q2.a);
    end
  end

endmodule
```

Output

```
# KERNEL: generated even number is  30
# KERNEL: generated even number is  12
# KERNEL: generated even number is  18
# KERNEL: generated even number is  24
# KERNEL: generated even number is  26
```

4.7. Question 7

- Constraint to generate distinct values for two variables.

```
class question3;
rand int a;
rand int b;

  constraint value{a!=b;
                   a inside{[0:100]};
                   b inside{[0:100]};
                  }

endclass

module answer3;
  question3 q3;

  initial begin
    q3=new();
    repeat(5)begin

      if(!q3.randomize())//checking randomize return 0 or 1 ; if 1
          success,0 false;
        $display("Constraint Failed");
      else
        $display("distinct value are %0d,%0d",q3.a,q3.b);
    end
  end

endmodule
```

Output

```
# KERNEL: distinct value are 95,7
# KERNEL: distinct value are 11,5
# KERNEL: distinct value are 90,2
# KERNEL: distinct value are 35,66
# KERNEL: distinct value are 28,25
```

4.8. Question 8

- Generate a 4-bit random value (0 to 15).

```
class question4;
rand bit[3:0] a;
endclass

module answer4;
  question4 q4;

  initial begin
    q4=new();
    repeat(5)begin

      if(!q4.randomize())//checking randomize return 0 or 1 ; if 1
          success,0 false;
        $display("Constraint Failed");
      else
        $display("4bit value are are %b | %0d",q4.a,q4.a);
    end
  end
endmodule
```

Output

```
# KERNEL: 4bit value are are 0001 | 1
# KERNEL: 4bit value are are 1000 | 8
# KERNEL: 4bit value are are 0100 | 4
# KERNEL: 4bit value are are 0011 | 3
# KERNEL: 4bit value are are 1000 | 8
```

4.9. Question 9

- Write a constraint so that dynamic array size is always 5.

```
class question5;
rand int arr[];

  constraint value {arr.size()==5;
                    foreach(arr[i])
                      arr[i] inside{[0:100]}; }
endclass

module answer5;
  question5 q5;

  initial begin
    q5=new();
    repeat(5)begin

      if(!q5.randomize())//checking randomize return 0 or 1 ; if 1
          success,0 false;
        $display("Constraint Failed");
      else
        $display("size of array is %d",q5.arr.size());
```

```
20          $display("value of array is %p",q5.arr);
21      end
22    end
23  endmodule
```

Output

```
# KERNEL: size of array is 5
# KERNEL: value of array is '{55, 5, 79, 17, 100}
# KERNEL: size of array is 5
# KERNEL: value of array is '{85, 85, 76, 36, 44}
# KERNEL: size of array is 5
# KERNEL: value of array is '{54, 98, 57, 98, 92}
# KERNEL: size of array is 5
# KERNEL: value of array is '{45, 6, 61, 79, 22}
# KERNEL: size of array is 5
# KERNEL: value of array is '{54, 72, 64, 70, 95}
```

4.10. Question 10

- Generate array elements in the range 0 to 9.

```
class question6;
rand int arr[10];

  constraint value{foreach(arr[i])
    arr[i]inside{[0:9]};}
endclass

module answer6;
  question6 q6;

  initial begin
    q6=new();
    repeat(5)begin

      if(!q6.randomize())//checking randomize return 0 or 1 ; if 1
         success,0 false;
        $display("Constraint Failed");
      else
        $display("generated value of array between 0 to 9 = %p",q6.arr)
           ;
    end
  end
endmodule
```

Output

```
1.generated value of array between 0 to 9 = '{5, 1, 2, 5, 0, 1, 4, 3, 2, 0}
2.generated value of array between 0 to 9 = '{9, 5, 3, 9, 5, 0, 9, 6, 7, 1}
3.generated value of array between 0 to 9 = '{3, 7, 7, 4, 9, 7, 0, 2, 6, 8}
4.generated value of array between 0 to 9 = '{0, 4, 6, 6, 6, 8, 1, 8, 8, 3}
5.generated value of array between 0 to 9 = '{1, 2, 4, 8, 3, 3, 8, 9, 4, 5}
```

4.11. Question 11

- Constraint: array should be in ascending order.

```
class question7;
  rand int arr[10];
  //int i=0;

  constraint value {
  foreach (arr[i])
     arr[i] inside {[1:200]};}

  //   function void ascending(); //using bubble sort for ascending
//      int temp;
//      for(int i=0;i<$size(arr);i++)
//        for(int j=0;j<$size(arr)-1;j++)
//          if(arr[j]>arr[j+1])begin
//            temp=arr[j];
//            arr[j]=arr[j+1];
//            arr[j+1]=temp;
```

```
//      end
//    endfunction
  constraint value2{foreach(arr[i])
    if(i>0)
      arr[i]>=arr[i-1];//using "<",">" for assigning ascending and
         decending ,here we use arr[i]>=arr[i-1] as arr[0]< arr[1]< arr
         [2] in ascending order use arr[i]<=arr[i-1]; for decending
         order for arr[0]> arr[1]> arr[2]
                    }
endclass


module answer7;
  question7 q7;

  initial begin
    q7=new();
    repeat(5)begin
      if(!q7.randomize())//checking randomize return 0 or 1 ; if 1
         success,0 false;

        $display("Constraint Fsailed");
      else
       // q7.ascending();
        $display("array in ascending order = %p", q7.arr);
    end
  end
endmodule
```

Output

```
# array in ascending order = '{37, 72, 79, 79, 83, 85, 96, 192, 196, 196}
# array in ascending order = '{35, 36, 61, 79, 90, 90, 98, 128, 137, 164}
# array in ascending order = '{1, 5, 11, 22, 47, 105, 106, 108, 140, 199}
# array in ascending order = '{15, 30, 30, 32, 59, 69, 74, 154, 154, 159}
# array in ascending order = '{22, 29, 78, 87, 130, 139, 140, 184, 187, 193}
```

4.12. Question 12

- Write constraint to generate only odd numbers in an array.

```
class question8;
  rand int arr [5];

  constraint value{foreach(arr[i]){
                    arr[i] inside{[0:200]};
                    arr[i]%2==1;
                    }
    unique{arr};}//using unique keyword to avoid repeated value
endclass

module answer8;
  question8 q8;
  initial begin
    q8=new();
    repeat(3)begin
```

```
    if(!q8.randomize())
      $display("Constraint error");
    else
      $display("The generated array of odd number is arr =%p",q8.arr);
  end
  end
endmodule
```

Output

```
# KERNEL: The generated array of odd number is arr ='{105, 73, 37, 175, 21}
# KERNEL: The generated array of odd number is arr ='{141, 139, 71, 121, 171}
# KERNEL: The generated array of odd number is arr ='{29, 153, 39, 49, 175}
```

4.13. Question 13

- Generate 1, 2, 3 but 1 should have highest weight.

```
class question9;
  rand int a;

  constraint value{
    a dist{1:=50,2:=20,3:=10};
    }

endclass

module answer9;
  question9 q9;
  initial begin
    q9=new();
    repeat(5)begin
      if(!q9.randomize())
      $display("Constraint error");
    else
      $display("The values are = %p",q9.a);
  end
  end
```

Output

```
# KERNEL: The values are = 2
# KERNEL: The values are = 1
# KERNEL: The values are = 1
# KERNEL: The values are = 1
# KERNEL: The values are = 1
```

4.14. Question 14

- Generate a number divisible by 4 and between 20–200

```
class question10;
  rand int a ;

  constraint value {a inside{[20:200]};
                    a%4==0;
                    }

endclass

module answer10;
  question10 q10;
  initial begin
    q10=new();
    repeat(5)begin
      if(!q10.randomize())
        $display("Constraint error");
      else
        $display("No:divisible by 4 between 20-200 =%0d",q10.a);
    end
  end
endmodule
```

Output

```
# KERNEL: No:divisible by 4 between 20-200 =188
# KERNEL: No:divisible by 4 between 20-200 =36
# KERNEL: No:divisible by 4 between 20-200 =60
# KERNEL: No:divisible by 4 between 20-200 =40
# KERNEL: No:divisible by 4 between 20-200 =112
```

4.15. Question 15

- Two variables should differ by at least 10.

```
class question11;
  rand int a;
  rand int b;

  constraint value{
    a inside{[0:100]};
    b inside{[0:100]};
  }
  constraint value2{a-b<=10;}

endclass

module answer11;
  question11 q11;
  initial begin
    q11=new();
    repeat(5)begin
      if(!q11.randomize())
        $display("Constraint error");
      else
        $display("value of a = %0d",q11.a);
        $display("value of b = %0d",q11.b);
        $display("--------------------------");
    end
  end
endmodule
```

Output

```
# KERNEL: value of a = 13
# KERNEL: value of b = 34
# KERNEL: --------------------------
# KERNEL: value of a = 22
# KERNEL: value of b = 36
# KERNEL: --------------------------
# KERNEL: value of a = 19
# KERNEL: value of b = 64
# KERNEL: --------------------------
# KERNEL: value of a = 17
# KERNEL: value of b = 42
# KERNEL: --------------------------
# KERNEL: value of a = 33
# KERNEL: value of b = 89
# KERNEL: --------------------------
```

4.16. Question 16

- Sum of all array elements should be less than 50

```
class question12;
  rand int arr[10];
  constraint value{
                   foreach(arr[i]){
                     arr[i] inside {[0:20]};
                   }
                     arr.sum<=50;
                     unique{arr};
                  }
endclass

module answer12;
  question12 q12;
  initial begin
    q12=new();
    repeat(5)begin
      if(!q12.randomize())
        $display("Constraint error");
      else
        $display("array = %p",q12.arr);
      $display("Sum of array = %0d",q12.arr.sum);
    end
  end
endmodule
```

Output

```
# KERNEL: array = '{3, 2, 7, 1, 9, 10, 8, 5, 4, 0}
# KERNEL: Sum of array = 49
# KERNEL: array = '{1, 2, 7, 4, 0, 3, 6, 5, 8, 12}
# KERNEL: Sum of array = 48
# KERNEL: array = '{14, 7, 8, 5, 2, 3, 6, 1, 0, 4}
# KERNEL: Sum of array = 50
# KERNEL: array = '{6, 2, 5, 8, 14, 3, 7, 4, 1, 0}
# KERNEL: Sum of array = 50
# KERNEL: array = '{2, 8, 0, 12, 6, 9, 1, 3, 4, 5}
# KERNEL: Sum of array = 50
```

4.17. Question 17

- First element of array fixed to 10, remaining elements random

```
class question13;
  rand int arr[10];

  constraint value1 {
    foreach(arr[i]){
    arr[i] inside {[0:100]};
      if(i==0)
        arr[i]==10;}
     }
endclass

module answer13;
  question13 q13;
  initial begin
    q13=new();
    repeat(5)begin
      if(!q13.randomize())
        $display("Constraint error");
      else
        $display("array = %p",q13.arr);
    end
  end
endmodule
```

Output

```
# KERNEL: array = '{10, 36, 100, 71, 3, 64, 95, 37, 34, 71}
# KERNEL: array = '{10, 56, 6, 22, 19, 77, 11, 25, 81, 51}
# KERNEL: array = '{10, 81, 88, 54, 21, 35, 32, 5, 43, 96}
# KERNEL: array = '{10, 56, 38, 38, 16, 39, 51, 63, 78, 76}
# KERNEL: array = '{10, 26, 2, 78, 93, 36, 68, 51, 76, 43}
```

4.18. Question 18

- If enable is 1, the value should be ¿ 100.

```
class question14;
  rand int enable;
  rand int a;

  constraint value {
    enable inside{[0:1]};
    enable dist{0:=1,1:=1};
    if(enable)
      a inside {[100:1000]};
    else
      a inside {[0:99]};
    }

endclass

module answer14;
  question14 q14;
  initial begin
    q14=new();
    repeat(5)begin
      if(!q14.randomize())
        $display("Constraint error");
      else
        $display("Value of enable=%0d",q14.enable);
        $display("the value of a %0d",q14.a);
      $display("-----------------------------");
    end
  end
endmodule
```

Output

```
# KERNEL: Value of enable=1
# KERNEL: the value of a 641
# KERNEL: -----------------------------
# KERNEL: Value of enable=1
# KERNEL: the value of a 243
# KERNEL: -----------------------------
# KERNEL: Value of enable=0
# KERNEL: the value of a 69
# KERNEL: -----------------------------
# KERNEL: Value of enable=0
# KERNEL: the value of a 1
# KERNEL: -----------------------------
# KERNEL: Value of enable=1
# KERNEL: the value of a 621
# KERNEL: -----------------------------
```

4.19. Question 19

- generate a pattern of 3,7,6,14,9,21,12,28....etc

```
class example;
  rand int arr[];//20

  constraint value {arr.size()==20;
                    foreach(arr[i])
                      if(i%2==0)
                        arr[i]==3*(i/2+1);
                    else
                      arr[i]==7*((i+1)/2);
                   }
endclass

module answer;
  example ex;
  initial begin
    ex=new();
    ex.randomize();
    $display("%p",ex.arr);
  end
endmodule
```

Output:

```
{3, 7, 6, 14, 9, 21, 12, 28, 15, 35, 18, 42, 21, 49, 24, 56, 27, 63, 30, 70}
```

4.20. Question 20

- generate pattern 0102030405 using constraint

```
class example;
  rand int arr[10];

  constraint c2 { foreach (arr[i])
    if(i%2==0)
      arr[i]==0;
    else
      arr[i]==(i+1)/2;
                                  }
endclass

module answer;
  example ex;
  initial begin
    ex=new();
    ex.randomize();
    $display("arr =%p",ex.arr);
  end

endmodule
```

output

```
arr ='{0, 1, 0, 2, 0, 3, 0, 4, 0, 5}
```