System Verilog and Constraints

1) Constraint for power of 9 or power of 3?

```
class power9;
  rand int a [];
constraint c1 {a.size == 10;}
 constraint c2 {foreach(a[i])
  if(i<10)
                  a[i] == (9**i);}
               //a[i] == (3**i);}
endclass
 module power_9;
  initial
    begin
      power9 p_h;
     p_h = new;
     p_h.randomize;
    $display("randomized data : %p", p_h.a);
  end
endmodule
```

2) Constraint for 11110000

```
Class pattern;
rand int a[];
 constraint c1 {a.size == 8;}
constraint c2 {foreach(a[i])
              if(i<4)
           //       if(i%2==0 || i%2==1 || i%2==1)
                     a[i] == 1;//}
//  constraint c3 {foreach(a[j])
                    // if(j>3)
                   //     if(j%4==0 || j%4==1 || j%4==2 || j%4==3)
                else
                         a[j] ==0;}
endclass
module power_3;
initial begin
 pattern p_h;
     p_h = new;
    p_h.randomize;
 $display("randomized data : %p", p_h.a);
  end
endmodule
```

3) constraint for {1000, 0100, 0010, 0001} diagonal

```
class matrix;

 rand int a[4][4];
  constraint c1 {foreach(a[i,j])
              if(i==j)
                   a[i][j] == 1;
            else
                 a[i][j] == 0;}
             /*   if(i==0)
                  if(j<1)
                    a[i][j] ==1;
                   else
                    a[i][j]==0;
                 else if(i==1)
                  if(j==1)
                    a[i][j] ==1;
                   else
                    a[i][j]==0;
                 else if(i==2)
                  if(j==2)
                    a[i][j] ==1;
                   else
                    a[i][j]==0;
                else
                  if(j==3)
                    a[i][j]==1;
                   else
                    a[i][j]==0;}*/
endclass
module diagonal_matrix;
  initial
    begin
     matrix m_h;
    m_h = new;
     m_h.randomize;
      $display("randomized data : %p", m_h.a);
  end
endmodule
```

4) constraint for {1111, 1110, 1100, 1000}

```
class matrix;
  rand int a[4][4];
```

```
  constraint c1 {foreach(a[i,j])
                 if(i==0)
                   if(j<4)
                     a[i][j] ==1;
                    else
                      a[i][j]==0;
                 else if(i==1)
                   if(j<3)
                     a[i][j] ==1;
                    else
                      a[i][j]==0;
                 else if(i==2)
                   if(j<2)
                     a[i][j] ==1;
                    else
                      a[i][j]==0;
                else
                  if(j<1)
                    a[i][j]==1;
                   else
                     a[i][j]==0;}
endclass
module diagonal_matrix;
initial
 begin
      matrix m_h;
     m_h = new;
     m_h.randomize;
      $display("randomized data : %p", m_h.a);
  end
endmodule
```

5) constraint for {1010,0101,1100,0011}

```
class matrix;
  rand int a[4][4];
  constraint c1 {foreach(a[i,j])
                 if(i==0)
                   if(j%2==0)
                     a[i][j] ==1;
                    else
                      a[i][j]==0;
                 else if(i==1)
                   if(j%2==0)
                     a[i][j] ==0;
```

```
                else
                  a[i][j]==1;
              else if(i==2)
                if(j<2)
                  a[i][j] ==1;
                else
                  a[i][j]==0;
              else
                if(j==3)
                  a[i][j]==1;
                else
                  a[i][j]==0;}
endclass
module diagonal_matrix;
  initial
    begin
      matrix m_h;
     m_h = new;
     m_h.randomize;
      $display("randomized data : %p", m_h.a);
  end
endmodule
```

6) constraint for pattern 123404321 or palindrome

```
    class pattern;
      rand int a[];
      constraint c1 { a.size==9;}
      constraint c2 {foreach(a[i])
                      if(i<4)
                        a[i] ==i+1;
                      else if(i==4)
                        a[i] ==0;
                      else if(i>4)
                        a[i] == 9-i;}
    endclass
    module patterns;
      initial
        begin
          pattern p1;
          p1 = new;
          p1.randomize;
          $display("randomized data : %p", p1.a);
        end
    endmodule
```

7) constraint for {0001, 0010,0100,1000}

```
class reverse_diagonal;
  rand int a[4][4];
  constraint c1 {foreach(a[i,j])
                 if(i==0 && j==3 || i==1 && j==2 || i==2 && j==1 || i==3 && j==0)
                    a[i][j]==1;
                  else
                    a[i][j]==0;}
endclass
module tb;
  initial
    begin
      reverse_diagonal p1;
      p1 = new;
      p1.randomize;
      $display("randomized data : %p", p1.a);
    end
endmodule
```

8) constraint for {1234,2341,3412,4123}

```
class matrix;
  rand int a[4][4];
constraint c1 {foreach(a[i,j])
               if(i==0)
                  a[i][j] == 1+j;
               else if(i==1)
                   if(j%2==0)
                     a[i][j] == 2+j;
                   else
                     a[i][j] == 4-j;
               else if(i==2)
                   if(j%2==0)
                     a[i][j] ==3-j;
                   else
                     a[i][j] == 5-j;
               else if(i==3)
                  if(j==0)
                     a[i][j] == 4;
                 else if(j>0 && j<4)
                    a[i][j] ==j;

  }
 endclass
module matrice;
```

```
  initial
    begin
      matrix m1;
      m1 = new;
      m1.randomize;
      $display("randomized data : %p", m1.a);
    end
endmodule
```

9) constraint for bandi 9966637002

```
class number;
  rand int a[];
  constraint c1 {a.size == 10;}
constraint c2 {foreach(a[i])
  if(i<2)
    a[i] == 9;
              else if(i>1 && i<5)
                a[i] ==6;
              else if(i%5==0)
                a[i] == 3;
              else if(i%4==2)
                a[i] == 7;
              else if ( i==7 || i==8)
                a[i] ==0;
              else if (i==9)
                a[i] == 2;
             }
endclass
module matrice;
  initial
    begin
      number m1;
      m1 = new;
      m1.randomize;
      $display("randomized data : %p", m1.a);
    end
endmodule
```

10)constraint for Fibonacci series

```
class fibonacci;
  rand int a[];
  constraint c1 {a.size == 10;}
constraint c2 { a[0] == 0; a[1] ==1;}
//  function void fibonacci_series;
 //   a[0] =0;
```

```
 //   a[1]=1;
//    for(int i=2;i<a.size; i++)
  //    a[i] = a[i-1]+a[i-2];
//  endfunction
Constraint c2 {foreach(a[i])
If(i>=2)
  a[i] = a[i-1]+a[i-2];}
endclass
module matrice;
  initial
    begin
      fibonacci m1;
      m1 = new;
      m1.randomize;
      m1.fibonacci_series;
      $display("randomized data : %p", m1.a);
    end
endmodule
```

11)constraint for reverse Fibonacci series

```
class fibonacci;
  rand int a[];
  constraint c1 {a.size == 10;}
  constraint c2 { a[0] == 34;
                  a[1] == 21;}
  constraint c3 {foreach(a[i])
    if(i>1)
    a[i] == a[i-2] - a[i-1];}
 /* function void fibonacci_series;
    a[0] =34;
    a[1]=21;
    for(int i=2;i<a.size; i++)
      a[i] = a[i-2]-a[i-1];
  endfunction */
endclass
module matrice;
  initial
    begin
      fibonacci m1;
      m1 = new;
      m1.randomize;
     //m1.fibonacci_series;
      $display("randomized data : %p", m1.a);
    end
```

```
endmodule
```

12)**constraint for two 4-bit variables such that in “a” variable, lsb bit should not equal to b variable lsb**.

```
class ab;
  rand bit [3:0] a;
  rand bit [3:0] b;
  constraint c1 { a[3:1] == b[3:1];}
endclass
module matrice;
  initial
    begin
      ab m1;
      m1 = new;
      //repeat(10)
      m1.randomize;
      //repeat(10)
      $display("randomized data : %b", m1.a);
      $display("randomized data : %b", m1.b);
    end
endmodule
```

13)constraint for '{'{2, 4, 6, 8}, '{1, 3, 5, 7}, '{2, 3, 4, 5}, '{1, 2, 2, 1}}

```
class matrix;
  rand int a[4][4];
  constraint c1 {foreach(a[i,j])
    if(i==0)
     // if(j%2==0)
        a[i][j] == 2+(j*2);
   else if(i==1)
     a[i][j] == 1+(j*2);
   else if(i==2)
     a[i][j] == i+j;
   else
     if(i==3 && j<=1)
       a[i][j] == 1+j;
              else if(i==3 && j==2 || i==3 && j==3)
       a[i][j] == 4-j;
             }
endclass
module matrice;
  initial
```

```
    begin
      matrix m1;
      m1 = new;
      m1.randomize;
      $display("randomized data : %p", m1.a);
    end
  endmodule
```

14) constraints for '{'{'{1, 2, 3, 4}, '{2, 3, 4, 1}, '{3, 4, 1, 2}, '{4, 1, 2, 3}}, '{'{2, 4, 6, 8}, '{1, 3, 5, 7}, '{2, 3, 4, 5}, '{1, 2, 2, 1}}}

```
  class matrix;
    rand int a[2][4][4];
    constraint c1 {foreach(a[i,j,k])
      if(i==0)
        if(j==0)
           a[i][j][k] == 1+k;
        else if(j==1)
          if(k%2==0)
            a[i][j][k] == 2+k;
              else
                a[i][j][k] == 4-k;
        else if(j==2)
          if(k%2==0)
            a[i][j][k] ==3-k;
          else
            a[i][j][k] == 5-k;
        else if(j==3)
          if(k==0)
            a[i][j][k] == 4;
          else if(k>0 && k<4)
            a[i][j][k] ==k;}
    constraint c2 {foreach(a[i,j,k])
      if(i==1)
        if(j==0)
          a[i][j][k] == 2+(k*2);
        else if(j==1)
          a[i][j][k] == 1+(k*2);
       else if(j==2)
        a[i][j][k] == j+k;
       else
        if(j==3 && k<=1)
          a[i][j][k] == 1+k;
        else if(j==3 && k==2 || j==3 && k==3)
          a[i][j][k] == 4-k;
```

```
                }
    endclass
    module matrice;
     initial
       begin
         matrix m1;
         m1 = new;
         m1.randomize;
         $display("randomized data : %p", m1.a);
       end
    endmodule
```

15) constraint for '{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

```
    class series;
     rand int a[];
     constraint c1 {a.size ==14;}
     constraint c2 {foreach(a[i])
        //if(i<15)
     //   if(i%2==0)
         a[i] ==2+i;
       // else
        // a[i] == i+2;
                }
    endclass
    module pattern;
     initial
       begin
         series s1;
         s1 = new;
         s1.randomize;
         $display("randomized data : %p", s1.a);
       end
    endmodule
# randomized data : '{'{1, 0, 2, 0}, '{0, 3, 0, 4}, '{5, 0, 6, 0}, '{0, 7, 0, 8}}
# exit
// Code your testbench here
// or browse
class matrix;
  rand int a[4][4];
  constraint c1 {foreach(a[i,j])
              if(i==0)
                if(j%2==0)
                  a[i][j] ==(j+2)/2;
                else
                  a[i][j]==0;
```

```
            else if(i==1)
              if(j%2==0)
                a[i][j] ==0;
              else
                a[i][j]== (6+j)/2;
            else if(i==2)
              if(j%2==1)
                a[i][j] ==0;
                else
                  a[i][j]==(10+j)/2;
          else
            if(j%2==0)
             a[i][j]==0;
               else
                 a[i][j]==(14+j)/2;
            }
endclass
module diagonal_matrix;
  initial
    begin
      matrix m_h;
     m_h = new;
     m_h.randomize;
      $display("randomized data : %p", m_h.a);
  end
endmodule
```

constraint for 001122334455

```
class pattern;
  rand int a[];
  constraint c1 {a.size == 12;}
  constraint c2 { foreach(a[i])
              a[i] == i/2;
              }
endclass
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
      p1.randomize;
      $display("randomize data : %p", p1.a);
```

```
    end
endmodule
```
constraint for 5*5 matrix such that last column is sum of previous columns
```
class matrix;
  rand int a[5][5];
  constraint c2 { foreach(a[i,j])
                  a[i][j] inside {[1:10]};}
  constraint c1 { foreach(a[i,j])
    if(j==4)
       a[i][4] == a[i][0]+a[i][1]+a[i][2]+a[i][3];
     /* else if(i==1)
       a[i][4] == a[i][0]+a[i][1]+a[i][2]+a[i][3];
      else if(i==2)
       a[i][4] == a[i][0]+a[i][1]+a[i][2]+a[i][3];
      else if(i==3)
       a[i][4] == a[i][0]+a[i][1]+a[i][2]+a[i][3];
      else
       a[i][4] == a[i][0]+a[i][1]+a[i][2]+a[i][3];*/
              }
endclass
module matrice;
  initial
    begin
      matrix m1;
      m1 = new;
      m1.randomize;
      $display("randomized data : %p", m1.a);
    end
endmodule
```
constraint for 1.45 to 7.43
```
class pattern;
  rand int a;
  real b;
  constraint c1 { a inside {[145:743]};}
  function void post_randomize();
    b = a/100.0;
    $display("random value : %f", b);
  endfunction
endclass
module tb;
  initial
    begin
      pattern p1;
```

```
      repeat(30)
        begin
      p1 = new;
      p1.randomize();
        end
    end
endmodule
```

constraint for pattern 888887777766666555544444333322222111110000

```
class pattern;
  rand int a[];
  constraint c1 {a.size==45;}
  constraint c2 {foreach(a[i])
    a[i] == 8 - (i/5);}
endclass
module tb;
  initial
    begin
      pattern p1;
      p1 =new;
      p1.randomize;
      $display("a = %p", p1.a);
    end
endmodule
```

constraint for pattern 9, 19, 29, 39,49,59,69

```
class packet;
  rand int a[8];
  constraint c {foreach(a[i])
    a[i] == (i*10)+9;}
endclass
module tb;
  packet p;
  initial begin
    p = new();
    p.randomize();
    $display( "pattern : %p",p.a);
  end
  endmodule
```

System Verilog Concepts

polymorphism

```
class polymorphism;
  function void display();
    $display("it is PARENT");
```

```
  endfunction
endclass
class extended extends polymorphism;
  virtual function void display();
    $display("it is CHILD");
  endfunction
endclass
module tb;
  initial
    begin
      polymorphism p1;
      extended e1 = new;
      p1 = e1;
      p1.display();
    end
endmodule
```

Associative array

```
module associative;
  int a[int];
   int c[string];
  string b[string];
  initial
    begin
      a = '{ 1 : 2025,
           5  : 34};

      c = '{ "age"    : 10,
           "salary" : 16000};

      b = '{"fruits" : "pomegranate",
           "vegetables" : "Tomato"};

      $display("a = %p ", a);
      $display("c = %p", c);
      $display("b = %p", b);
    end
endmodule
```

inheritance

```
class base_class;
  bit [2:0] a;
endclass
class extend_class1 extends base_class;
  bit [3:0] b;
```

```
  int c;
endclass
module tb;
  initial
    begin
      base_class b=new;
      extend_class1 e1=new;
      b.c=2; // illegal access
      e1.a= 3'd4;
      e1.b=4'd5;
      e1.c= 1;
      $display("randomized data :%p, randomized data :%p, randomized data :%p",
e1.a,e1.b,e1.c);
    end
endmodule
```

associate array with methods

```
// Code your testbench here
// or browse Examples
module associative();
  int a[int];
  int id;
  initial
    begin
      a[3]=5;
      a[1]= 2;
      a[25] = 24;
      a[3000] =1;
      if(a.exists(25))
        $display("entry exists in mem, whose value is %d", a[25]);
      else
        $display("no entry");
       if(a.prev(id))
       $display("previous entry %d is made in address %d", a[id], id);
      else
         $display("no entry");
      if(a.first(id))
        $display("first entry %d is made in address %d", a[id], id);
      else
         $display("no entry");
      if(a.last(id))
        $display("last entry %d is made in address %d", a[id], id);
      else
         $display("no entry");
```

```
      $display("number of entries in array is %0d", a.num);
    end
endmodule
```

pass by ref

```
module argument_passing;
 int x,y,z;
 //function to add two integer numbers.
 function automatic int sum(ref int x,y);
 x = x+y;
 return x+y;
 endfunction
 initial begin
 x = 20;
 y = 30;
 z = sum(x,y);
 $display("-----------------------------------------------------");
 $display("\tValue of x = %0d",x);
 $display("\tValue of y = %0d",y);
 $display("\tValue of z = %0d",z);
 $display("-----------------------------------------------------");
 end
endmodule
```

logical gates

```
`timescale 1ns/1ns
module example();
  reg [3:0] a;
  reg [3:0] b;
  initial
    begin
      a=4'b0011; b=4'b1101;
      $display ("a = %0b,\n b = %0b ,\na&&b=%0b, \na||b=%0b , \na&b=%0b
,\na|b=%0b ", a, b, a&&b, a||b, a&b, a|b);
        $dumpfile();     $dumpfile("wave1.vcd");
    end
 // $display ("a = %0b and b = %0b", a, b, a&&b, a||b, a&b, a|b);
Endmodule
```

Static casting

```
module static_casting();
  real r;
  int a;
  initial
    begin
      r = (1.8*3.2);
```

```
      a = int'(r);
      $display(" r = %f", r);
      $display(" a = %d", a);
    end
endmodule
```

shallow copy

```
class dummy;
  int a;
endclass
class main;
  dummy d_h = new();
  int d;
endclass
module tb;
  initial
    begin
      main m1, m2;
      m1 = new;
      m1.d = 10;
      m1.d_h.a=11;
      $display("a = %0d, d = %0d", m1.d, m1.d_h.a);
      //shallow copy
      m2 = new m1;
      $display("a = %0d, d = %0d", m2.d, m2.d_h.a);
      $display("modification");
      m2.d_h.a =20;
      $display("a = %0d, d = %0d", m1.d, m1.d_h.a);
      $display("a = %0d, d = %0d", m2.d, m2.d_h.a);
    end
endmodule
```

deep copy

```
class dummy;
  int a;
  function dummy copy();
    copy = new();
    copy.a = this.a;
    return copy;
  endfunction
endclass
class main;
  dummy d_h = new();
  int d;
  function main copy();
```

```
    copy = new;
    copy.d = this.d;
    copy.d_h = this.d_h.copy;
    return copy;
  endfunction
endclass
module tb;
  initial
    begin
      main m1, m2;
      m1 = new;
      m1.d = 10;
      m1.d_h.a=11;
      $display("a = %0d, d = %0d", m1.d, m1.d_h.a);
      //deep copy
      m2 = m1.copy();
      $display("a = %0d, d = %0d", m2.d, m2.d_h.a);
      $display("modification");
      m2.d_h.a =20;
      $display("a = %0d, d = %0d", m1.d, m1.d_h.a);
      $display("a = %0d, d = %0d", m2.d, m2.d_h.a);
    end
endmodule
```

constraint for Armstrong number

```
class armstrong;
  rand int a;
  constraint c1 {a inside { 153, 370, 371, 407 };}
  function void post_randomize();
    int r, temp, sum;
    temp = a;
    while(a >0)
      begin
        r = a % 10;
        sum = (r**3)+sum;
        a = a/10;
      end
    if(temp == sum)
      $display("it is armstrong number is %0d", temp);
    else
      $display("it is not a armstrong number is %0d", temp);
  endfunction
endclass
```

```
module tb;
  initial
    begin
      armstrong a_h;
      a_h = new;
      a_h.randomize;
    end
endmodule
```

semaphore

```
module tb();
  semaphore sem;
  task display();
    sem.get(1);
    #5;
    $display("process 1",$time);
    sem.put(1);
  endtask
  task display1();
    sem.get(1);
    #4
    $display("process 2",$time);
    sem.put(1);
  endtask
  task display2();
    sem.get(1);
    #2
    $display("process 3",$time);
    sem.put(1);
  endtask
  initial
    begin
      sem = new(1);
      fork
        display();
        display1();
         display2();
      join
    end
endmodule
```

constraint for mobile number such that first 4 numbers are 8919

```
class mobile;
```

```
  rand int a[];
  constraint c1 { a.size == 10;}
  constraint c2 {foreach(a[i]) a[i] inside {[0:9]};}
  constraint c3 {foreach(a[i])
    if(i==1 ||  i==3)
      a[i] == 9;
                else if( i== 0)
                  a[i] == 8;
                else if(i==2)
                 a[i] == 1;}
endclass
module tb;
  initial
    begin
      mobile m;
      repeat(10)
        begin
          m = new;
          m.randomize;
          $display("mobile number : %p", m.a);
        end
    end
endmodule
```

pattern 1010110101

```
class pattern;
  rand int a[10];
  constraint c1 { foreach(a[i])
    if(i<5)
      if(i%2==0)
        a[i] == 1;
      else
        a[i] == 0;
              }
  constraint c2 { foreach(a[i])
    if(i>4)
      if(i%2==0)
        a[i] == 0;
      else
        a[i] == 1;
              }
endclass
module tb;
```

```
  initial
    begin
      pattern p_h;
      p_h = new;
      p_h.randomize;
      $display("randomized data : %p", p_h.a);
    end
endmodule
```

pattern 11101110

```
class pattern;
  rand int a[8];
  constraint c1 { foreach(a[i])
    if(i<3 || i>3 && i<7)
      a[i] == 1;
                else
                  a[i] ==0;
               }
endclass
module tb;
  initial
    begin
      pattern p_h;
      p_h = new;
      p_h.randomize;
      $display("randomized data : %p", p_h.a);
    end
endmodule
```

9,99,999,9999,99999

```
class pattern;
  rand int a[9];
  constraint c1 { foreach(a[i])
    if(i==0)
      a[i] ==9;
                else
                   a[i] == 9 + 10 * a[i-1] ;  }
endclass
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
```

```
    p1.randomize;
    $display(" %p", p1.a);
  end
endmodule
```

dynamic array with methods

```
module dyn_example;
  int a[];
  initial
    begin
      a = new[10];
      a = '{10,20,30,40,50,60,70,80,90,100};
      foreach(a[i])
        $display("a[%0d] = %0d", i, a[i]);
      $display("size of array = %0d", a.size);
      a = new[25] (a);
      foreach(a[i])
        $display("a[%0d] = %0d", i, a[i]);
      $display("size of array = %0d", a.size);
      a= new[20];
      foreach(a[i])
        $display("a[%0d] = %0d", i, a[i]);
      $display("size of array = %0d", a.size);
    end
  endmodule
```

palindrome

```
class palindrome;
  rand int a;
  constraint c1 {a inside { [100:999]};}
  function void post_randomize();
    int r, temp, sum;
    temp = a;
    while(a >0)
      begin
        r = a % 10;
        sum = sum*10+r;
        a = a/10;
      end
    if(temp == sum)
      $display("it is palindrome number is %0d", temp);
    else
      $display("it is not a palindrome number is %0d", temp);
  endfunction
```

```
endclass

module tb;
  initial
    begin
      palindrome a_h;
      a_h = new;
      repeat(600)
        begin
          a_h.randomize;
        end
    end
endmodule
```

000111222333

```
class pattern;
  rand int a[12];
  constraint c1 { foreach(a[i])
    a[i] == i/3; }
endclass
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
      p1.randomize;
      $display("randomize data : %p", p1.a);
    end
endmodule
```

constraint with unique keyword

```
class unique_ex;
  rand bit [3:0] a[10];
constraint c2 { unique {a};}
endclass
module tb;
  initial
    begin
      unique_ex u_h;
      u_h = new;
```

```
        u_h.randomize;
        $display("unique random data : %p", u_h.a);
    end
endmodule
```

constraint for odd numbers in even locations and even numbers in odd location

```
class odd_even;
  rand int a[10];
  constraint c2 { foreach(a[i])  a[i] inside {[10:20]};}
  constraint c1 { foreach(a[i])
    if(i%2==0)
      a[i]%2==1;
              else
                a[i]%2==0;}
endclass
module tb;
  initial
    begin
      odd_even o_h;
      o_h = new;
      o_h.randomize;
      $display("randomized data : %p", o_h.a);
    end
endmodule
```

25,27,30,36,40,45

```
class pattern;
  rand int a[7];
  constraint c1 { foreach(a[i]) a[i]>24;}
  constraint c3 { foreach(a[i]) a[i]<46;}
  constraint c2 { foreach(a[i])
    (a[i]%9==0) || (a[i]%5==0);}
  constraint c4 { foreach(a[i]) a[i]!=35;}

endclass
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
        p1.randomize;
        $display("a = %p", p1.a);
    end
endmodule
```

123123123123

```
class patt_gen;
  rand int a[];
  constraint c1 { a.size==12;}
  constraint c2 { foreach(a[i])
    a[i] == (i%3)+1;
                }
endclass
module tb;
  initial
    begin
      patt_gen p1;
      p1 = new;
      p1.randomize;
      $display("a=%p", p1.a);
    end
endmodule
```

11001100110011001100

```
class pattern;
  rand int a[20];

  constraint c1 { foreach(a[i])
    if((i/2)%2==0)
      a[i] == 1;
               else
                 a[i] == 0;}
endclass
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
      p1.randomize;
      $display("randomize data : %p", p1.a);
    end
endmodule
```

1,22,3,33,5,44,7,55

```
class pattern;
  rand int a[8];
  constraint c1 { foreach(a[i])
    if(i%2==0)
      a[i] == i+1;
                else
```

```
                a[i] == 11*(i+3)/2;
              }
endclass
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
      p1.randomize;
      $display("a = %p", p1.a);
    end
endmodule
```

1,11,3,22,5,33,7,44,9,55

```
class pattern;
  rand int a[10];
  constraint c1 { foreach(a[i])
    if(i%2==0)
      a[i] == i+1;
              else
                a[i] == 11*(i+1)/2;
              }
endclass
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
      p1.randomize;
      $display("a = %p", p1.a);
    end
endmodule
```

1,22,3,44,5,66,7,88,7,66,5,44,3,22,1

```
class pattern;
  rand int a[15];
  constraint c1 { foreach(a[i])
    if(i%2==0 && i<8)
      a[i] == i+1;
              else if(i%2==1 && i<8)
                a[i] == 11*(i+1);
              else
                a[i] == a[14-i];
              }
endclass
```

```
module tb;
  initial
    begin
      pattern p1;
      p1 = new;
      p1.randomize;
      $display("a = %p", p1.a);
    end
endmodule
```