```
//<------------J_V_M_DURGA_PHANITEJA_MANNIDI---------->//

//Q1.Write a constraint to generate below pattern 0101010101...
//ANS:
class cons;
      rand int a[];
      constraint x{a.size==10;}
      constraint y{foreach(a[i])
                              if(i%2==0)
                                    a[i]==0;
                              else
                                    a[i]==1;}
      function void post_randomize();
            $display("Randomized data is %0p",a);
      endfunction
endclass

cons c;
module top;
      initial
            begin
                  c=new();
                  assert(c.randomize());
            end
endmodule

//Q2:Write a constraint to generate below pattern 1234554321
//ANS:
class cons;
      rand int a[];
      constraint x{a.size==10;}
      constraint y{foreach(a[i])
                              if(i<5)
                                    a[i]==i+1;
                              else
                                    a[i]==10-i;}
      function void post_randomize();
            $display("Randomized data is %0p",a);
      endfunction
endclass

cons c;
module top;
      initial
            begin
```

```
                        c=new();
                        assert(c.randomize());
                end
endmodule

//Q3:Write a constraint to generate below pattern 9 19 29 39 49 59 69 79
//ANS:
class cons;
        rand int a[];
        constraint x{a.size==7;}
        constraint y{foreach(a[i])
                                        a[i]==(i*10)+9;}
        function void post_randomize();
                $display("Required pattern is %0p",a);
        endfunction
endclass

cons c;

module top;
        initial
                begin
                        c=new();
                        assert(c.randomize());
                end
endmodule

//Q4:write a constraint to generate below pattern 5 -10 15 -20 25 -30
//ANS:
class cons;
        rand int a[];
        constraint size{a.size==6;}
        constraint x{foreach(a[i])
                                        if(i%2==0)
                                                a[i]==(i*5)+5;
                                        else
                                                a[i]==-5*(i+1);}
        function void post_randomize();
                $display("Required pattern is %0p",a);
        endfunction
endclass

cons c;

module top;
        initial
```

```
            begin
                  c=new();
                  assert(c.randomize());
            end
endmodule

//Q5:Write a constraaint to generate the pattern 1122334455
//ANS:
class cons;
      rand int da[];
      constraint size{da.size==10;}
      constraint x{foreach(da[i])
                              da[i]==(i+2)/2;}
      function void post_randomize();
            $display("Required pattern is %0p",da);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new();
                  assert(c.randomize);
            end
endmodule

//Q6:Write a code to generate random number between 1.35 and 2.57
//ANS:
class cons;
      rand int a;
      real b;
      constraint size{a inside {[1350:2570]};}
      function void post_randomize();
            b=a/1000.0;
            $display("Random number in between 1.35 and 2.57 is %0f",b);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new();
                  repeat(10)
                  assert(c.randomize);
            end
endmodule

//Q7:Write a constraint to generate the pattern 0102030405.
//ANS:
class cons;
      rand int a[];
      constraint size{a.size==10;}
      constraint x{foreach(a[i])
                        if(i%2==0)
                              a[i]==0;
                        else
                              a[i]==(i+2)/2;}
      function void post_randomize();
            $display("Required pattern is %0p",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new();
                  assert(c.randomize);
            end
endmodule

//Q8:Write constraint to generate random values 25,27,30,36,40,45
without using "set membership".
//ANS:
class cons;
      rand int a;
      constraint x{a>24;a<46;}
      constraint y{(a%5==0)||(a%9==0);a!=35;}
      function void post_randomize();
            $display("Random value is %0d",a);
      endfunction
endclass

cons c;

module top;
      initial
```

```
            begin
                  c=new();
                  repeat(10)
                        assert(c.randomize);
            end
endmodule

//Q9:Write a constraint to generate a random even number between 50 and
100.
//ANS:
class cons;
      rand int even[];
      constraint range{even.size==51;}
      constraint x{foreach(even[i])
                              even[i] inside {[50:100]};}
      constraint y{foreach(even[i])
                              even[i]%2==0;}
      function void post_randomize();
            $display("Random even numbers between 50 and 100 are:\n
%0p",even);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new();
                  assert(c.randomize());
            end
endmodule

//Q10:Write a for a 32 bit rand variable such that it should have 12
number of 1's non consecutively.
//ANS:
class cons;
      rand bit [31:0]a;
      constraint no_of_ones{$countones(a)==12;}
      //We can use constraint no_of_ones{$countbits(a,1)==12;}
      constraint x{foreach(a[i])
                              if(i>0&&a[i]==1)
                                    a[i]!=a[i-1];}
      function void post_randomize();
            $display("Randomized 32 bit variable having 12 ones
is:%0b",a);
```

```
            $display("Displaying no of ones in the variable that we
randomized is %0d",$countones(a));
      endfunction
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//Q11:Write a constraint to genarate FACTORIAL of first 5 even numbers
and odd numbers .
//ANS:
class cons;
      rand int even[];
      rand int odd[];
      //function for doing the factorial
      function int fact(int i);
            if(i==0)
                  fact=1;
            else
                  fact=i*fact(i-1);
      endfunction
      //constraint
      constraint size{even.size==5;}
      //for even numbers factorial
      constraint even_fact{foreach(even[j])
                                          even[j]==fact(2*(j+1));}
      //for odd numbers factorial
      constraint odd_fact{foreach(odd[k])
                                          odd[k]==fact((2*k)+1);}
      function void post_randomize();
            $display("Factorial of 1st five even numbers is %0p",even);
            //$display("Factorial of 1st five odd numbers is %0p",odd);
      endfunction
endclass

cons c;

module test;
      initial
```

```
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//Q12: Write a constraint such that even locations contains odd numbers
and odd locations contains even numbers.
//ANS:
class cons;
      rand int eo[];
      constraint size{eo.size inside {[10:20]};}
      constraint range{foreach(eo[i])
                              eo[i] inside {[1:100]};}
      constraint x{foreach(eo[i])
                        if(i%2==0)
                              eo[i]%2==1;
                        else
                              eo[i]%2==0;}
      function void post_randomize();
            $display("After randomizing data is %0p",eo);
      endfunction
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//Q13: Write a SystemVerilog program to randomize a 32-bit variable, but
only randomize the 12th bit.
//ANS:
class cons;
      rand bit[31:0]a;
      constraint x{foreach(a[i])
                        if(i==12)
                              a[i] inside {[0:1]};
                        else
                              a[i]==0;}
      function void post_randomize();
            $display("After randomizing value of a is %0b",a);
      endfunction
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//Q14:Write a constraint on two dimensional array for generating even
numbers in the first 4 locations and odd numbers in the next 4
locations.Also the even number should be in multiple of 4 and odd number
should be multiple of 3.
//ANS:
class cons;
      rand int a[2][4];
      constraint x{foreach(a[i])
                              foreach(a[i][j])
                                    if(i==0)
                                          a[i][j]%4==0;
                                    else
                                          a[i][j]%2!=0 && a[i][j]%3==0;}
      constraint y{foreach(a[i])
                              foreach(a[i][j])
                                    a[i][j] inside {[1:100]};}

      function void post_randomize();
            $display("Randomized value is %0p",a);
      endfunction
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  assert(c.randomize);
            end
endmodule

//Q15:Constraint to generate bit[7:0] array1[10] with unique values and
```

```
also multiple of 3.
//Ans:
class cons;
      rand bit [7:0]array1[10];
      constraint y{foreach(array1[i])
                              array1[i]%3==0 && array1[i] inside
{[1:100]};}
      constraint z{unique{array1};}
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  assert(c.randomize);
                  $display("%0p",c.array1);
            end
endmodule

//Q16:Write a constraint to generate unique numbers in an array without
using unique keyword.
//ANS:
class cons;
      rand int a[10];
      constraint x{foreach(a[i])
                              foreach(a[j])
                                    if(i!=j)
                                          a[i]!=a[j];}
      constraint y{foreach(a[i])
                              a[i] inside {[1:100]};}
      function void post_randomize();
            $display("Unique numbers are %0p",a);
      endfunction
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule
```

```
//Q18:Write a constraint to generate prime numbers between the range of
1 to 100.
//Ans:
class cons;
      rand int a;
      int b;
      constraint range{a inside {[1:100]};}

      function int prime(int a);
            if(a<=1)
                  return 2;
            for(int i=2;i<a;i++)
                  begin
                        if(a%i==0)
                              return 3;
                        else
                              prime=a;
                  end
      endfunction

      function void post_randomize;
            b=prime(a);
            $display("Prime number is %0d",b);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  repeat(20)
                  assert(c.randomize);
            end
endmodule

//Q19:Write a constraint to generate a variable with 0-31 bits should be
1, 32-61 bits should be 0.
//Ans:
class cons;
      rand bit [0:61]a;
      constraint x{foreach(a[i])
                        if(i<32)
                              a[i]==1;
                        else
                              a[i]==0;}
      function void post_randomize();
            $display("Randomized value is %0b",a);
      endfunction
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  assert(c.randomize);
            end
endmodule

//Q20:Write a constraint to generate consecutive elements using Fixed
size array and also write the constraint to generate non consecutive
elements?
//ANS:
class cons;
      rand int a[10];
      //for consecutive elements
      constraint x{foreach(a[i])
                        a[i]==i;}
      //for non consecutive elements
      constraint y{foreach(a[i])
                        foreach(a[j])
                              if(i!=j)
                                    a[i]!=a[j];}
      constraint z{foreach(a[i])
                        a[i] inside {[0:100]};}

      function void post_randomize();
            $display("Conseutive values are %0p",a);
      endfunction
endclass
cons c;
module test;
      initial
            begin
                  c=new;
                  assert(c.randomize);
            end
endmodule
```

```
//Q21:write a constraint to randmoly genrate 10 unquie numbers between
99 to 100.
//Ans:
class cons;
      rand int a[10];
      real b[10];
      constraint x{foreach(a[i])
                              a[i] inside {[990:1000]};}
      constraint z{foreach(a[i])
                              foreach(a[j])
                                    if(i!=j)
                                          a[i]!=a[j];}

      function void post_randomize();
            foreach(b[i])
            b[i]=a[i]/10.0;
            $display("10 unique numbers between 99 and 100 are:%0p",b);
      endfunction
endclass
cons c;
module test;
      initial
            begin
                  c=new;
                  assert(c.randomize);
            end
endmodule

//Q22:Write a constraint sunch that array size between 5 to 10 values of
the array are in asscending order.
//In this program Iam going to use extern constraint.
//Ans:
class cons;
      rand int a[];
      constraint size;
      constraint ascending;

      function void post_randomize();
            $display("After randomizing a is %0p",a);
      endfunction
endclass

constraint cons::size{a.size inside {[5:10]};}
constraint cons::ascending{foreach(a[i])
                                          a[i] inside {[1:10]};
```

```
                                        foreach(a[i])
                                           if(i>0)
                                                 a[i]>a[i-1];}
cons c;

module top;
      initial
            begin
                  c=new;
                  assert(c.randomize);
            end
endmodule

//Q23:Write a constraint to generate even numbers should in the range of
10 to 30 using Fixed size array, Dynamic array & Queue.
//Ans:
class cons;
      rand int a[10];//fixed size array
      rand int b[];//dynamic array
      rand int c[$];//queue

      constraint sizeb{b.size inside {[5:10]};}
      constraint sizec{c.size inside {[5:10]};}

      constraint x{foreach(a[i])
                                    a[i] inside {[10:30]} && a[i]%2==0;
                               foreach(b[i])
                                    b[i] inside {[10:30]} && b[i]%2==0;
                               foreach(c[i])
                                    c[i] inside {[10:30]} && c[i]%2==0;}

      function void post_randomize();
            $display("a is %0p \n b is %0p \n c is %0p",a,b,c);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule
```

```
//Q24:Write a constraint so that if we randomize a single bit variable
for 10 times values should generate be like 101010101010.
//Ans:
class cons;
      rand bit a;
      bit b=0;
      constraint x{a!=b;}
      function void post_randomize();
            $write("%0b%0b",a,b);
      endfunction
endclass

cons c;
module top;
      initial
            begin
                  c=new;
                  repeat(5)
                  c.randomize;
            end
endmodule

//Q25:Write a constraint to demonstrate solve before constraint.
//Ans:
class cons;
      rand bit a;
      rand bit [3:0] b;
      constraint x{(a==0)->(b==1);solve a before b;}
      function void post_randomize();
            $display("a is %0d and b is %0d",a,b);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  repeat(20)
                  c.randomize;
            end
endmodule


//26.Write a code to generate unique elements in an array without using
```

```
unique keyword and constraints.
//Ans:
module uniquee;
      int a[10];
      initial
            begin
                  foreach(a[i])
                        a[i]=i+1;
                  a.shuffle();
                  $display("%0p",a);
            end
endmodule

//27.Write a constraint for 2D dynamic array to print consecutive
elements.
//Ans:
class cons;
      rand int a[][];
      constraint size1{a.size inside {[10:20]};
                        foreach(a[i])
                              a[i].size==2;}
      constraint s{foreach(a[i])
                        foreach(a[i][j])
                              a[i][j]==i+j;}
      function void post_randomize();
            $display("Consecutive elements in two_d array are %0p",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  assert(c.randomize);
            end
endmodule

//28.Constraint to check whether the randomized number is palindrome or
not
//Ans:
//Palindrome_number:A palindrome number is a number that remains the
same when digits are reversed.
class cons;
      rand int num;
      constraint x{num inside {[100:999]};}

      function void post_randomize();
            int r,sum,temp;
                  temp=num;
                  //As we are checking the 3 digit number is palindrome
or not,so we have 3 iterations here.
                  for(int i=0;i<3;i++)
                  begin
                  r=num%10;
                  sum=(sum*10)+r;
                  num=num/10;
                  end
                  if(temp==sum)
                        $display("%0d is a palindrome number",temp);
                  else
                        $display("%0d is not a palindrome number",temp);

      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  repeat(20)
                  c.randomize;

            end
endmodule

//29.Write a constraint to display the fibonacci sequence.
//Ans:
//Fibonacci sequence:The next number is found by adding up the two
numbers before it
class cons;
      rand int n;
      rand int a[];
      constraint x{n inside {[7:15]};}
      constraint y{a.size==n;}
      constraint z{foreach(a[i])
                        if(i==0)
                              a[i]==0;
                        else if(i==1)
```

```
                                    a[i]==1;
                              else
                                    a[i]==a[i-1]+a[i-2];}
      function void post_randomize();
            $display("Fibonacci sequence is %0p",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//30.Write a code to check whether the randomized number is an armstrong
number or not.
//Ans:
//Armstrong number:An Armstrong number is one whose sum of digits raised
to the power three equals the number itself.For example 371 is an
Armstrong number because 3**3 + 7**3 + 1**3 = 371.
class cons;
      rand int num;
      constraint x{num inside {[100:999]};}
      function void post_randomize();
            int temp,sum,r;
            temp=num;
            for(int i=0;i<3;i++)
                  begin
                        r=num%10;
                        sum=(r**3)+sum;
                        num=num/10;
                  end
            if(temp==sum)
                  $display("%0d is an armstrong number",temp);
            else
                  $display("%0d is not an armstrong number",temp);
      endfunction
endclass

cons c;

module top;
```

```
      initial
            begin
                  c=new;
                  repeat(20)
                  c.randomize();
            end
endmodule

//Q31:Write a constraint so that the elements in two queues are
different.
//Ans:
module test;
      class AC;
      rand int q_1[$];
      rand int q_2[$];
      constraint distinct{q_1.size inside {[10:20]};
                              q_2.size inside {[10:20]};
                              foreach(q_1[i])
                                    q_1[i] inside {[1:100]};
                              foreach(q_2[i])
                                    !(q_2[i] inside {q_1});}
      function void post_randomize();
            $display("q2 is %0p and q1 is %0p",q_2,q_1);
      endfunction
      endclass
      initial
            begin
                  AC a=new;
                  a.randomize;
            end
endmodule

//Q32:Constraint for 0-100 range is 70% and 101-255 range is 30%.
//Ans:
class cons;
      rand bit [7:0]a;
      constraint x{a dist {[0:100]:=70,[101:255]:=30};}
      function void post_randomize();
            $display(a);
      endfunction
endclass

cons c;

module top;
      int count;
      initial
            begin
                  c=new;
                  repeat(100)
                  begin
                        c.randomize;
                        if(c.a<101)
                              count=count+1;
                  end
                  $display("No of entries less than 100 are %0d",count);
                  $display("No of entries less than 100 are
%0d",100-count);
            end
endmodule

//Q33:Write a constraint for 16-bit variable such that no two
consecutive ones should be generated.
//Ans:
class cons;
      rand bit[32:0]a;
      constraint x{foreach(a[i])
                              if(i>0 && a[i]==1)
                                    a[i]!=a[i-1];}
      function void post_randomize();
            $display("%0b",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  c.randomize();
            end
endmodule

//Q34:Write a constraint to generate 32 bit number with 1 bit high using
$onehot().
//Ans:
//$onehot(expression):It returns true if exactly one bit of expression
is high
class cons;
      rand bit[31:0]a;
      constraint x{$onehot(a);}
      function void post_randomize;
            $display("%0b",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//Q35:Write a constraint to randomly generate unique prime numbers in an
array between 1 and 200. The generated prime numbers should have 7 in
it.
//Ans:
class cons;
      rand int a[];
      int q[$];
      constraint size{a.size inside {[100:150]};}
      constraint x{foreach(a[i])
                        a[i]==prime(i);}

      function int prime(int j);
            if(j<=1)
                  return 2;//here we can return any prime value
            for(int k=2;k<j;k++)
                  begin
                        if(j%k==0)
                              return 3;//here we can return any prime
value
                        else
                              prime=j;
                  end
      endfunction

      function void post_randomize();
            foreach(a[i])
                  begin
                  if(a[i]%10==7)
                        q.push_front(a[i]);
                  end
                  $display("%0p",q);
```

```
	endfunction
endclass

cons c;

module top;
	initial
		begin
			c=new;
			c.randomize;
		end
endmodule

//Q36:Write a constraint on a 16 bit rand vector to generate alternate
pairs of 0's and 1's.
//Ans:Ex:0011.. or 1100..
class cons;
	rand bit [15:0]a;
	constraint x{foreach(a[i])
				if(i<15)
				if(i%2==0)
					a[i+1]==a[i];
				else
					a[i]!=a[i+1];}
	function void post_randomize;
		$display("%0b",a);
	endfunction
endclass

cons c;

module top;
	initial
		begin
			c=new;
			c.randomize;
		end
endmodule

//Q37.Assume that you have two properties
	//rand bit[7:0]a and rand bit[2:0]b
	//The range of values for b are 3,4,7
	//if b==3,the number of ones in the 'a' must be 4.
	//if b==4,the number of ones in the 'a' must be 5.
	//if b==7,the number of ones in the 'a' must be 8.
//Ans:
```

```
class cons;
      rand bit[7:0]a;
      rand bit[2:0]b;
      constraint x{b dist {3:/2,4:/2,7:/6};}
      constraint y{$countones(a)==b+1;}
      //or we can use
      //constraint z{$countbits(a,1)==b+1;}
      function void post_randomize;
            $display("b=%0d-->number of 1's in 'a' is
%0d",b,$countones(a));
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  repeat(10)
                        c.randomize;
            end
endmodule

//Q38.Write a constraint such that sum of any three consecutive elements
in an array should be an even number.
//Ans:
class cons;
      rand int a[];
      constraint x{a.size inside {[10:20]};}
      constraint y{foreach(a[i])
                              if(i>1)
                                    (a[i]+a[i-1]+a[i-2])%2==0;}
      constraint z{foreach(a[i])
                              a[i] inside {[0:10]};}
      function void post_randomize;
            $display("array is %0p",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
```

```
                        c.randomize;
                end
endmodule

//Q39:Write a constraint for two random variables such that one variable
should not match with the other & the total number of bits toggled in
one variable should be 5 w.r.t the other.
//Ans:
class cons;
        rand bit [8:0]a;
        rand bit [8:0]b;
        constraint not_equal{a!=b;}
        constraint count{$countones(a)==5;}
        function void post_randomize;
                $display("a is %0b\n b is %0b",a,b);
        endfunction
endclass

cons c;

module top;
        initial
                begin
                        c=new;
                        c.randomize;
                end
endmodule


//Q40.Write a constraint such that when rand bit[3:0] a is randomized,
the value of "a" should not be same as 5 previous occurrences of the
value of "a".
//Ans:
class cons;
        rand bit [3:0]a;
        int que[$];
        constraint x{!(a inside {que});}

        function void post_randomize;
                que.push_front(a);
                if(que.size==6)
                        que.pop_back;
                $display("a is %0d",a);
        endfunction

endclass
```

```
cons c;

module top;
      initial
            begin
                  c=new;
                  repeat(8)
                  c.randomize;
            end
endmodule

//Q41.Write a code to have the cyclic randomization behaviour without
using the randc keyword.
//Ans:
class cons;
      rand bit [1:0] a;
      int que[$];
      constraint x{!(a inside {que});}

      function void post_randomize();
            que.push_front(a);
            if(que.size==2**$size(a))
                  begin
                        que={};
                  end
            $display("a is %0d",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  repeat(8)
                  c.randomize;
            end
endmodule

//Q42:Write constraint for payload such that the size should be randomly
generated in between 11 and 22 and each value of the payload should be
grater than previous value by 2.
//ANS:
class cons;
```

```
      rand int payload[];
      constraint size{payload.size inside {[11:22]};}
      constraint y{foreach(payload[i])
                              payload[i] inside {[0:100]};}
      constraint x{foreach(payload[i])
                              if(i>0)
                                    payload[i]==payload[i-1]+2;}
      function void post_randomize();
            $display("After randomizing payload data is %0p",payload);
      endfunction
endclass

cons c;

module test;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//43.Write a constraint to print unique elements in a 2D array without
using unique keyword.
//Ans:
class cons;
      rand int a[][];
      constraint x{a.size inside {[5:10]};
                        foreach(a[i])
                              a[i].size inside {[1:3]};}
      constraint y{foreach(a[i])
                              foreach(a[i][j])
                                    foreach(a[m])
                                          foreach(a[m][n])
                                    if(!(i==m && j==n))
                                          a[i][j]!=a[m][n];}
      constraint z{foreach(a[i])
                              foreach(a[i][j])
                                    a[i][j] inside {[0:100]};}
      function void post_randomize;
            foreach(a[i])
                  begin
                        foreach(a[i][j])
                              $display("a[%0d][%0d] is %0d",i,j,a[i][j]);
                  end
      endfunction
```

```
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule

//44.Sorting the elements in the dynamic array using constraints.
//Ans:
class cons;
      rand int a[];
      constraint size{a.size inside {[10:20]};}
      constraint range{foreach(a[i])
                                    a[i] inside {[0:100]};}
      function void post_randomize;
            $display("Before sorting array is %0p",a);
            foreach(a[i])
                  begin
                        for(int j=i+1;j<$size(a);j++)
                              begin
                                    if(a[i]>a[j])
                                          begin
                                                a[i]=a[i]+a[j];
                                                a[j]=a[i]-a[j];
                                                a[i]=a[i]-a[j];
                                          end
                              end
                  end
            $display("After sorting array is %0p",a);
      endfunction
endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule
```

```
//Q45:Write a constraint to generate 1221122112211.....
//Ans:
class cons;
      rand int a[];
      constraint x{a.size inside {[10:20]};}
      constraint y{foreach(a[i])
                              if(i%4==0 || i%4==3)
                                    a[i]==1;
                              else
                                    a[i]==2;}
      function void post_randomize;
            $display("Required pattern is %0p",a);
      endfunction

endclass

cons c;

module top;
      initial
            begin
                  c=new;
                  c.randomize;
            end
endmodule
```