SYSTEM VERILOG

CONSTRAINTS – Part 4

Write a constraint for an array of size 10, if any number comes as 0 in between then the next number must be 1.

```
class constraint_31;
	rand int da[];
	constraint c1{da.size==10;}
	constraint c2{foreach(da[i])
				da[i] inside {[0:5]};}
	constraint c3{foreach(da[i])
				if (i<9 && da[i] == 0)
					da[i+1] == 1;}
endclass

constraint_31 c1;

module test();
	initial
		begin
			repeat(10)
				begin
					c1=new;
					assert(c1.randomize());
					$display("da: %p",c1.da);
				end
		end
endmodule
```

Generate a palindrome of length 8 where values are random between 1 to 5.

```
class constraint_32;
	rand int da[];
	constraint c1{da.size == 8;}
	constraint c2{foreach(da[i])
				da[i] inside {[1:5]};}
	constraint c3{foreach(da[i])
				da[i] == da[da.size() - 1 - i]; }
endclass

constraint_32 c1;

module test();
	initial
		begin
			repeat(10)
				begin
					c1=new;
					assert(c1.randomize());
					$display("da: %p",c1.da);
				end
		end
endmodule
```

Randomize a time value on HH:MM:SS 24-hour format using constraints. Ensure all fields are valid.

```
class constraint_33;
	rand bit[4:0]hour;
	rand bit[5:0]minute;
	rand bit[5:0]second;
	constraint c1{hour inside {[0:23]};}
	constraint c2{minute inside {[0:59]};}
	constraint c3{second inside {[0:59]};}
endclass

constraint_33 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("%d:%d:%d",c1.hour,
c1.minute, c1.second);
				end
		end
endmodule
```

Write a constraint that generates a random date in the format "DD.MM.YYYY" ensuring leap year handling.

```
class constraint_34;
	rand int date,month,year;
	constraint c1{month inside {[1:12]};}
	constraint c2{
		if (month inside {1,3,5,7,8,10,12})
			date inside {[1:32]};
		else if (month == 2)
			{if ((year%4 == 0 && year%100 != 0) || (year % 400 == 0))
				date inside {[1:30]};
			else
				date inside {[1:28]}; }
		else
			date inside {[1:30]}; }
	constraint c3{year inside {[1000:3000]};}
	endclass

constraint_34 c1;
```

```
module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("%d.%d.%d",c1.date,
c1.month, c1.year);
				end
		end
endmodule
```

Create a constraint random Boolean truth table (8-bits) ensuring only one '1' appears.

```
class constraint_35;
	rand bit [7:0]y;
	constraint c1{$countones(y) == 1;}
endclass

constraint_35 c1;
```

```
module test();
	initial
		begin
			repeat(10)
				begin
					c1=new;
					assert(c1.randomize());
					$display("y=%b",c1.y);
				end
		end
endmodule
```

Write a class that generates a sequence of increasing numbers randomly between 0 to 50.

```
class constraint_36;
	rand int da[];
	constraint c1{da.size == 10;}
	constraint c2{foreach(da[i])
				da[i] inside {[0:50]};}
	constraint c3{foreach(da[i])
				if(i>0)
					da[i] > da[i-1]; }
endclass
```

```
constraint_36 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("da: %p",c1.da);
				end
		end
endmodule
```

Randomize an array of size 10 integers such that no two consecutive numbers are equal.

```
class constraint_37;
	rand int da[];
	constraint c1{da.size == 10;}
	constraint c2{foreach(da[i])
				da[i] inside {[0:20]};}
	constraint c3{foreach(da[i])
				if(i>0)
					da[i] != da[i-1]; }
endclass
```

```
constraint_37 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("da: %p",c1.da);
				end
		end
endmodule
```

Randomize a phone number pattern: ‘ +91 XXXXXXXXXX ’ ensuring first digit after +91 is non-zero.

```
class constraint_38;
	rand int number[];
	constraint c1{number.size == 10;}
	constraint c2{number[0] != 0;}
	constraint c3{foreach(number[i])
				number[i] inside {[0:9]};}
```

```
	function void print();
		$display("+91-%d %d %d %d %d %d %d %d %d %d", number[0], number[1], number[2], number[3], number[4], number[5], number[6], number[7], number[8], number[9]);
	endfunction
endclass

constraint_38 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					c1.print();
				end

		end
endmodule
```

Write a constraint to generate an array where sum of even indexed elements equal to sum of odd indexed elements.

```
class constraint_39;
	rand int da[];
	constraint c1{da.size == 10;}
	constraint c2{foreach(da[i])
				da[i] inside {[0:20]};}
	constraint c3{(da[0] + da[2] + da[4] + da[6] + da[8])
== (da[1] + da[3] + da[5] + da[7] + da[9]);}
endclass

constraint_39 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("da: %p",c1.da);
				end
		end
endmodule
```

Write a constraint to generate a random array of size 4 such that the sum of the elements is equal to 10, the randomization should only generate numbers between 0 to 9.

```
class constraint_40;
	rand int da[];
	constraint c1{da.size == 4;}
	constraint c2{foreach(da[i])
				da[i] inside {[0:9]};}
	constraint c3{da.sum() == 10;}
endclass

constraint_40 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("da: %p",c1.da);
				end
		end
endmodule
```