1. **Write a Verilog code to design a clock with period = 100 ns and a duty cycle of 25% by using always and initial statements. The value of clock at time = 0 should be initialized to 0.**

```
`timescale 1ns/1ns

`define CLK_PERIOD 100

module reg_tb();
        reg CLK=1'b0;
        always begin
                if(CLK) begin
                        #(` CLK_PERIOD /4) CLK = ~CLK;
                end
                else begin
                        #(` CLK_PERIOD *3/4) CLK = ~CLK;
                end
        end



        initial begin
        $vcdpluson;
        #500 $finish;
        end

endmodule
```

2. A 4-variable logic function that is equal to 1 of any three or all four its input variables are equal to 1 is called a *majority* function.

a. Write a *gate-level* Verilog module with four inputs and one output that implements the majority function using the basic gate primitives and, or, xor, not.

```
`timescale 100ps/100ps

module majority(out,a,b,c,d);
input a,b,c,d;
output out;
wire a1,a2,a3,a4;

and and1(a1,a,b,c);
and and2(a2,a,b,d);
and and3(a3,a,c,d);
and and4(a4,b,c,d);

or or1(out,a1,a2,a3,a4);

endmodule
```

b. Write a *dataflow* Verilog module with the same inputs and outputs that implements the same majority function using continuous assignment statements.

```
`timescale 100ps/100ps

module majority(out,a,b,c,d);
input a,b,c,d;
output out;
wire a1,a2,a3,a4;

assign a1=a&b&c;
assign a2=a&b&d;
assign a3=a&c&d;
assign a4=b&c&d;

assign out=a1|a2|a3|a4;

endmodule
```

3. Given values of *a*, *b*, and *c* as shown, write the result of expressions shown below.

Assume: a is [3:0], b is [3:0], c is [5:0]
Assume: a = 4'b0010 , b = 4'b1010 , c = 6'b001101

Evaluate the following expressions:

a & b = ?	**a || b = ?**
a && b = ?	**a | b = ?**
a + b = ?	**a = c , a = ?**
a - b = ?	**c = b , c = ?**
&b = ?	**| a = ?**

a & b = 0010

a || b = 0001

a && b = 0001

a | b = 1010

a + b = 1100

a=c, a= 1101

a - b = 1000

c=b, c= 001010

& b = 0000

| a = 0001

4. What will be the output after running the following initial blocks?

```
//Initial block A
initial
   begin

     x = 1;
      $display ("Hello, x=%d, t=%t", x, $time);
      #15;

      x = 2;
      $display ("Hello, x=%d, t=%t", x, $time);
      #20;

      x = 3;
      $display ("Hello, x=%d, t=%t", x, $time);
      #10;

    end

  // Initial block B
  initial
    begin
      #5;

      x = 10;
      $display ("Hello, x=%d, t=%t", x, $time);
      #20;

      x = 20;
      $display ("Hello, x=%d, t=%t", x, $time);
      #5;

      x = 30;
      $display ("Hello, x=%d, t=%t", x, $time);
      #10;

    End
```

```
Hello, x= 1, t=                    0
Hello, x=10, t=                    5
Hello, x= 2, t=                   15
Hello, x=20, t=                   25
Hello, x=30, t=                   30
Hello, x= 3, t=                   35
```

Initil Block A	
x	**t**
1	0
10	5
2	15
20	25
30	30
3	35