1. **Adja meg az alábbi portokkal rendelkező Verilog modul modul deklarációját. Bemenetek: skalár clk, skalár en; kimenet: 8 bites dout.**

```
module mymodule(
  input   clk,
  input   en,
  output  [7:0]   dout
);
//…
endmodule
```

2. **Deklaráljon egy 4 bites data nevű, wire típusú jelet, és folyamatosan adja neki a hexadecimális 0x42 értéket bináris formában megadva.**

```
wire   [3:0]   data;
assign data = 8'b01000010;
```

3. **Deklaráljon három darab 8 bites, előjeles wire típusú változót (res, op_a, op_b). A res értékének adja át op_a és op_b összegét.**

```
wire  signed  [7:0]   res, op_a, op_b;
assign res = op_a+op_b;
```

4. **Deklaráljon egy 16 bites és egy 8 bites wire típusú változót (d16, d8). A 16 bites változónak adja át a 8 bites változó előjel kiterjesztett értékét (a 8 bites változó kettes komplemens értékeket tartalmazhat).**

```
wire   [15:0]   d16;
wire   [7:0]   d8;
assign d16={8{d8[7]}, d8};
```

5. **Deklaráljon egy 8 bites, reg típusú (res) és két darab 8 bites wire típusú változót (op_a, op_b). Adja meg azt a Verilog kódot, ami olyan kombinációs logikát valósít meg, amelyben res értéke op_a és op_b összege.**

```
reg   [7:0]   res;
wire   [7:0]   op_a, op_b;

always @ (op_a, op_b)
  res <= op_a+op_b;
```

6. **Adja meg egy 4:1 multiplexer Verilog kódját. Bemeneti jelek: kiválasztó jel (sel), adat bemenetek (in0, in1, in2, in3); kimenet: r. A bemeneti jeleket deklarálja wireként (az adatok 1 bitesek), a kimenetet pedig a leírás módjának megfelelően.**

```
module mux_41(
  input   [1:0]   sel,
  input   in0, in1, in2, in3,
  output   reg   r
);

always @ (*)
  case(sel)
    2'b00: r <= in0;
    2'b01: r <= in1;
    2'b10: r <= in2;
    2'b11: r <= in3;
  endcase
endmodule
```

7. **Adja meg egy aszinkron reset-elhető (rst), set-elhető (set), engedélyezgető (en) 1 bites D FF Verilog kódját. Deklarálja a kimeneti és bemeneti jeleket is (utóbbiakat 1 bites wire-ként).**

```
module d_ff(
  input   d, clk, rst, set, en,
  output   q
);

reg   out;
always @ (posedge clk, posedge rst, posedge set)
  if(rst)
    out <= 0;
  else if(set)
    out <= 1;
  else if (en)
    out <= d;

assign q = out;   //wire csak assign-al kaphat értéket, always blokkban nem

endmodule
```

8. **Adja meg azon testbench kódját, ami 10 MHz-es órajelet generál. Az órajel neve legyen clk, ezt deklarálja is.**

```
'timescale 1ns / 1ps
module tb_akarmi;

reg clk;

initial
   clk = 0;

always #50 //10MHz esetén T= 100ns
   clk <= ~clk;

endmodule
```

9. **Adja meg egy 8 bites, reset-elhető, felfelé számláló számláló modul Verilog kódját. Bemenetek: órajel (clk), reset (rst); kimenet: a számláló értéke (cntr).**

```
module cntr_8(
   input    clk, rst,
   output    [7:0]    cntr
);

reg    [7:0]    c;
always @ (posedge clk)    //feltételeztem, hogy szinkron reset
   if(rst)
      c <= 0;
   else
      c <= c+1;    //ha túlcsordul, pont jól alakul

assign cntr = c;    // wire csak assign-al kaphat értéket, always blokkban nem

endmodule
```

10. **Adja meg egy 8 bites, balra léptető shift regiszer modul Verilog kódját. Bemenetek: órajel (clk), soros adatbemenet (din); kimenet: a shift regiszter értéke (shr).**

```
module    sh_l8(
   input    clk,
   input    din,
   output    [7:0]    shr
);

reg    [7:0]    dout;
always @ (posedge clk)
   dout <= {din[6:0], din};

assign shr = dout;    // wire csak assign-al kaphat értéket, always blokkban nem

endmodule
```