Q1:

a)

```
module comp_1 (input [7:0] A, output [7:0] W);
   assign W = ~A + 1'b1;
endmodule
```

b)

```
module comp_1_b (input [7:0] A, output [7:0] W);

   genvar  i;
   wire [7:0] C;

   // Half adder
   assign W[0] = ~A[0] ^ 1'b1;
   assign C[0] = ~A[0] & 1'b1;

   generate
      for (i = 1; i < 8; i = i + 1) begin
         assign W[i] = ~A[i] ^ C[i - 1];
         assign C[i] = ~A[i] & C[i - 1];
      end
   endgenerate

endmodule
```

Q2:

a)

```
module adder_a (input a, b, ci, output sum, co);
   assign {co, sum} = a+b+ci;
endmodule
```

c)

```
module full_adder_c (input [3:0] a, b, input ci, output [3:0] w, output co);
  wire [4:0] wc;
  assign wc[0] = ci;
  genvar i;

  generate
    for (i = 0; i < 4; i = i + 1) begin
    adder_a myadder(.a(a[i]), .b(b[i]), .ci(wc[i]), .co(wc[i + 1]), .sum(w[i]));
    end
  endgenerate

  assign co = wc[4];
endmodule
```

Q4:

a)

```
module 3_to_8_dec (input [2:0] A, input en, output [7:0] W);
  assign w = (en == 1'b1) ? 8'b0:
    (A == 3'b000) ? 8'b00000001:
    (A == 3'b001) ? 8'b00000010:
    (A == 3'b010) ? 8'b00000100:
    (A == 3'b011) ? 8'b00001000:
    (A == 3'b100) ? 8'b00010000:
    (A == 3'b101) ? 8'b00100000:
    (A == 3'b110) ? 8'b01000000:
    (A == 3'b111) ? 8'b10000000:
    8'bx;
Endmodule
```

Q6:

a)

```
module _4_to_1_MUX_l (input [3:0] A, input [1:0] S, input en, output W);
    assign W = (en == 1'b1) ? 1'bz:
        (S == 2'b00) ? A[0]:
        (S == 2'b01) ? A[1]:
        (S == 2'b10) ? A[2]:
        (S == 2'b11) ? A[3]:
        1'bx;
endmodule

module _4_to_1_MUX_h (input [3:0] A, input [1:0] S, input en, output W);
    assign W = (en == 1'b0) ? 1'bz:
        (S == 2'b00) ? A[0]:
        (S == 2'b01) ? A[1]:
        (S == 2'b10) ? A[2]:
        (S == 2'b11) ? A[3]:
        1'bx;
endmodule
```

b)

```
module _16_to_1_MUX (input [15:0] A, input [3:0] S, input en, output W);
    wire [3:0] wr;
    _4_to_1_MUX_h mux0(.A(A[3:0]), .S(S[1:0]), .en(en), .W(wr[0]));
    _4_to_1_MUX_h mux1(.A(A[7:4]), .S(S[1:0]), .en(en), .W(wr[1]));
    _4_to_1_MUX_h mux2(.A(A[11:8]), .S(S[1:0]), .en(en), .W(wr[2]));
    _4_to_1_MUX_h mux3(.A(A[15:12]), .S(S[1:0]), .en(en), .W(wr[3]));
    _4_to_1_MUX_h mux4(.A(wr), .S(S[3:2]), .en(en), .W(W));
Endmodule
```

c)

```
module func_c (input [4:0] S, output W);


    wire e = S[0];

    _16_to_1_MUX mux (.A({1'b1, e, ~e, ~e, ~e, ~e, e, e, 1'b1, 1'b0, ~e, 1'b1, 1'b0, 1'b1, e, e}), .S(S[4:1]),
.en(1'b1), .W(W));


Endmodule
```

Q7:

c)

```
module comprator_1 (input a , b , l , g , e , output lt , eq , gt);


    assign lt = l | (e & ~a & b);

    assign eq = (~a & ~b & e) | (a & b & e);

    assign gt = g | (e & a & ~b);


endmodule
```

d)

```
module comprator_8 (input [7:0] a , b, input l , g , e , output lt , eq , gt);


   genvar i;


   wire [7:0] wl, wg, we;


   comprator_1 m(.a(a[7]), .b(b[7]), .l(l), .g(g), .e(e), .lt(wl[7]), .gt(wg[7]), .eq(we[7]));
   generate
      for (i = 6; i >= 0; i = i - 1) begin
         comprator_1 mi(.a(a[i]), .b(b[i]), .l(wl[i + 1]), .g(wg[i + 1]), .e(we[i + 1]), .lt(wl[i]), .gt(wg[i]),
.eq(we[i]));
      end
   endgenerate


   assign {lt, eq, gt} = {wl[0], we[0], wg[0]};


endmodule
```

Q8:

c)

```
module comprator_2_bit (input [1:0] a , b , input l , g , e , output lt , eq , gt);


   assign eq = (e & ~a[1] & ~b[1] & ~a[0] & ~b[0]) | (e & ~a[1] & ~b[1] & a[0] & b[0]) | (e & a[1] & b[1] &
~a[0] & ~b[0]) | (e & a[1] & b[1] & a[0] & b[0]);
   assign gt = g | (e & a[1] & ~b[1]) | (e & ~a[1] & ~b[1] & a[0] & ~b[0]) | (e & a[1] & b[1] & a[0] & ~b[0]);
   assign lt = l | (e & ~a[1] & b[1]) | (e & ~a[1] & ~b[1] & ~a[0] & b[0]) | (e & a[1] & b[1] & ~a[0] & b[0]);
endmodule
```

d)

```
module comprator_8 (input [7:0] a , b, input l , g , e , output lt , eq , gt);

   wire [8:0] GT , EQ , LT;

    assign {GT[8] , EQ[8] , LT[8]} = {g , e , l};
    genvar i;
    generate
        for(i = 0 ; i < 8 ; i = i + 1)begin
            comprator_2_bit comp({a[i+1] , a[i]} , {b[i+1],b[i]} , LT[i+2] , GT[i+2] , EQ[i+2] , LT[i] , EQ[i] , GT[i]);
        end
    endgenerate
    assign {lt , eq , gt} = {LT[0] , EQ[0] , GT[0]};
endmodule
```