Note: problems 1 and 2 are a little trickier than last semester! Read carefully!

1) Given $F = A'B' + CD'$
   a. Write $F'$ in product of sums notation
      \[
      F' = \overline{\overline{A'B'} + CD'} = \overline{\overline{A'B'}} \cdot \overline{CD'} = (A + B)(C + D')
      \]
   b. Implement $F$ using as few 2 input NOR gates as possible. Assume that only the true literals ($A, B, C, D$) are available, not their complements ($A', B', C', D'$).

   ![NOR gate diagram](image)

2) Given $G = (A' + B')(C + D')$
   a. Write $G'$ in sum of products notation.
      \[
      G' = \overline{(A' + B')(C + D')} = \overline{A'} \cdot \overline{B'} \cdot \overline{C} \cdot \overline{D} = \overline{A B + C D}
      \]
   b. Implement $G$ using as few 2 input NAND gates as possible. Assume that only the true literals are available, not their complements.

   ![NAND gate diagram](image)
3) Answer the following questions for the FSM below:
   a. Briefly describe the function of this sequence detector. When is the output 1?

   The seq det outputs a 1 when it receives a 1 as input that was not immediately preceded by a 0 3 and a 0 otherwise.

   b. Write a Verilog module which would implement this FSM for input variable "In" and output variable "Out." Use the same standard format as was presented in the Lab 3 lecture and used in Lab 3. (Define your states; use one always block for next state and output; use one always block for state transition)

```verilog
module FSM (Reset, Clock, In, Out);
    input Reset, Clock, In;
    output Out;
    reg [1:0] curr;
    reg [0:0] next;
    reg Out;
    parameter Star = 2'b00;
    parameter Ray = 2'b01;
    parameter Mo = 2'b10;
    always @ (posedge Clock) begin
        if (Reset)
            curr <= Star;
        else
            curr <= next;
    end always
    always @ (curr or In) begin
        next = curr; // default = stay here
        case (curr)
            Star: begin
                Out = 1'b0;
                if (In)
                    next = Ray; // stay here for In
            end // Star
            Ray: begin
                Out = 2'b01;
                if (In)
                    next = Mo;
                else
                    next = Star;
            end // Ray
            Mo: begin
                Out = 1'b0;
                if (!In)
                    next = Star; // stay here for In
            end // Mo
        endcase
    end always
endmodule // FSM
```
4) A finite state machine has one input and one output. The output becomes 1 and remains 1 thereafter when at least two 0s and at least two 1s have occurred as inputs, in any order after reset. Draw a state diagram of this FSM as a Moore machine. Try to minimize the number of states.

5) A Moore machine has one input and one output. The output should be 1 if the total number of 0s at the input is odd, and the total number of 1s at the input is an even number greater than 0. Draw a state diagram. Try to minimize the number of states.
6) Design a 3-Flip-Flop counter which transitions through states $Q_2Q_1Q_0 = 000, 100, 110, 111, 011, 001$ and then repeats.

a. Draw the state diagram and state transition table
b. Draw the Karnaugh maps, clearly indicating the implicants that you use in your covers of the next-state functions.
c. Implement the counter using D flip flops and whatever gates you like.
d. Is your counter self-starting? If yes, show the transitions of the unused states. If no, change it to make it self-starting, and show the transitions of the unused states.

\[ Q_2' = Q_0 \]
\[ Q_1' = Q_2 \]
\[ Q_0' = Q_1 \]

To fix: change $Q_1$ from $Q_2 + Q_0Q_1$ to $Q_2 + Q_0Q_1'$. Since $Q_1$ changed, then $Q_0$ is invalid. Leave state transitions for invalid states.

Now it's self-starting:

\[
\begin{array}{cccccc}
Q_2 & Q_1 & Q_0 & Q_2' & Q_1' & Q_0' \\
0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0
\end{array}
\]
Assume all gates have exactly the same delay.