Problem 1. A logic network has three inputs (Count, Input1, Input0) and two outputs (Output1, Output0):

The function implemented by $F$ is the following: if Count=0, then O1-O0 becomes I1-I0. If Count=1, then O1-O0 becomes (I1-I0)+1. For example, if Count=1 and I1-I0 is 01, O1-O0 becomes 10. Note that if I1-I0 is 11 and Count=1, then the output value wraps around 00, without any overflow.

a) Fill the Truth Table. 

<table>
<thead>
<tr>
<th>Count</th>
<th>I1</th>
<th>I0</th>
<th>O1</th>
<th>O0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

b) Write the Boolean expression for O1 and O0 in canonical sum of products form.

$$O1 = \overline{CNT}.I1.I0 + \overline{CNT}.I1.I0 + \overline{CNT}.I1.L0 + \overline{CNT}.I1.L0$$

$$O0 = \overline{CNT}.I1.I0 + \overline{CNT}.I1.I0 + \overline{CNT}.I1.L0 + \overline{CNT}.I1.L0$$
c) Fill in the K-Maps for Output1 (O1) and Output0 (O0), and find the Boolean expression for the minimum sum of products implementation.

\[ O1 = \overline{CNT}.I1 + I1.\overline{I0} + \overline{CNT}.\overline{I1}.I0 \]

\[ O0 = CNT.\overline{I0} + \overline{CNT}.I0 \]

d) Draw the schematics (gate netlists) for O1 and O0. Transform the netlists to NAND only schematics.
Problem 2. Given the function $F(A, B, C, D)$ as follows:

$$F = \overline{ABC} + \overline{ABC} + \overline{ABD} + \overline{ABD} + \overline{ACD} + \overline{ACD} + \overline{BCD}$$

Function is implemented by the following circuit:
For this multiplexer select signals, \( S_0 \) is the most significant bit and \( S_1 \) is the least significant bit. In equation (I) through (IV), select (circle) the function that you would use to implement \( F_0 \) to \( F_3 \). Use the following choices to answer each subquestion.

a) \( Z = X \text{ xor } Y \)  
   b) \( Z = X \text{ xnor } Y \)  
   c) \( Z = X \text{ and } Y \)  
   d) \( Z = X \text{ nand } Y \)  
   e) \( Z = X \text{ or } Y \)  
   f) \( Z = X \text{ nor } Y \)

(I) \( F_0 \) should be:  
   (a)   (b)   (c)   (d)   (e)   (f)  
   (II) \( F_1 \) should be:  
   (a)   (b)   (c)   (d)   (e)   (f)  
   (III) \( F_2 \) should be:  
   (a)   (b)   (c)   (d)   (e)   (f)  
   (IV) \( F_3 \) should be:  
   (a)   (b)   (c)   (d)   (e)   (f)  

\[ \begin{array}{ccc}
0 & 0 & 1 \overline{D} + \overline{B}C \\
0 & 1 & CD \\
1 & 0 & C + D \\
1 & 1 & \overline{C} + \overline{D}
\end{array} \]  

- \( F_0 \) is \text{ xor}  
- \( F_1 \) is \text{ and}  
- \( F_2 \) is \text{ or}  
- \( F_3 \) is \text{ nand}
Problem 3. Design a simplified traffic-light controller that switches traffic lights on a crossing where a north-south (NS) street intersects an east-west (EW) street. The input to the controller is the WALK button pushed by pedestrians who want to cross the street. The outputs are two signals NS and EW that control the traffic lights in the NS and EW directions.

When NS or EW are 0, the red light is on, and when they are 1, the green light is on. When there are no pedestrians, NS=0, EW=1 for a minute, follow by NS=1 and EW=0 for 1 minutes, and so on, when WALK button is pushed, NS and EW both become 0 for a minute when the present minute expires. After that the NS and EW signals continue alternating.

For this traffic-light controller:

a) Develop a state diagram. (Hint: can be done using 3 states)

b) Draw the state transition table.

c) Encode the states using minimum number of bits.

d) Derive the logic schematic for a sequential circuit which implements the state transition table.
Problem 4. Design hardware that implements the following pseudo-code using the provided Comparator, Adder and Registers, along with as many multiplexers and demultiplexers as needed. The comparator has two inputs \( \text{In1} \) and \( \text{In2} \), and three outputs, \( \text{C1} \), \( \text{C2} \), and \( \text{C3} \). If \( \text{In1} < \text{In2} \), \( \text{C1} = 1 \); if \( \text{In1} = \text{In2} \), \( \text{C2} = 1 \); if \( \text{In1} > \text{In2} \), \( \text{C3} = 1 \) (for a given \( \text{In1} \) and \( \text{In2} \), only one of the comparator outputs can be 1). The Adder takes as inputs two numbers \( p \) and \( q \), and produces an output \( \text{Sum} \). There are 5 registers for storing the 5 variables, \( A, B, X, Y, \) and \( Z \).

- Hint: You do not need to use truth table or K-maps. Insert the muxes/demuxes as appropriate, and show the signal connections from the input registers \( A, B, X \) to the output registers \( Y \) and \( Z \), through the muxes, comparator, adder, and demuxes. Be sure to show the equations for the select lines of the multiplexers/demultiplexers in terms of the comparator outputs, \( \text{C1}, \text{C2}, \) and \( \text{C3} \).

**Pseudo-code:**

If \( A < B \) then
\[
Z = X + A
\]

Else if \( A = B \) then
\[
Z = X + B
\]

Else
\[
Y = A + B
\]

Diagram: [Diagram of the circuit with labels and connections]