1.) For the following diagram, use the laws of Boolean Algebra to minimize this circuit. You do not have to name the laws used, but each step must be obvious to someone who knows the laws. Your solution should be in the form of a Boolean Equation.

\[
A \cdot B + \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{D}
\]

\[
A \cdot B + \overline{B} \cdot C + \overline{B} \cdot D + \overline{D}
\]
2.) Simplify the following K-maps, and write the corresponding Boolean Equation. You should create the simplest Sum of Products form possible.

\[ CD + \overline{A}B \overline{D} \]
3.) Draw the state diagram for the following circuit. YOU DO NOT HAVE TO DO THE STATE TABLE. YOU DO NOT HAVE TO IMPLEMENT THE CIRCUIT.

Design a circuit that has a 1-bit input. It outputs 1 every time when the last three input values seen (2 previous + the current) are the same, and 0 otherwise. At startup assume that the two previous input values were 1’s.
4.) For the following state diagram, implement the circuit. The state encoding is given to you in the parentheses in each state. You may use premade D-flipflops and any other basic gates. Your implementation should be as simple as possible.

```
<table>
<thead>
<tr>
<th>PS IN</th>
<th>Out NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

\[ NS = PS + \overline{IN} \]

\[ Out = \overline{PS} \overline{IN} \]
5.) For the Verilog given below, draw the corresponding finite state machine. DO NOT IMPLEMENT THE CIRCUIT. The code below is correct, legal Verilog.

```verilog
t
module machine (clk, reset, in, out);
  input logic clk, reset, in;
  output logic out;

  logic [1:0] ps;
  logic [1:0] ns;

  always_comb begin
    ns[0] = (ps[0] & ps[1]) | in;
    out = ps[1];
  end

  always_ff @(posedge clk) begin
    if (reset)
      ps <= 2'b11;
    else
      ps <= ns;
  end
endmodule
```

**State Diagram:**

![State Diagram](image)