1.) Using the rules of Boolean Algebra, minimize this function, showing the steps you use. You do not have to name the laws used, but each step must be obvious to someone who knows the laws.

\[ V = C + (AB + \overline{AE}) + E(\overline{C} + \overline{D}) + \overline{A} \]

\[ = C + (\overline{A} + \overline{B})(A + \overline{E}) + \overline{C}E + D\overline{E} + \overline{A} \]

\[ = \overline{C} \overline{E} + \overline{A} + \overline{E} + \overline{A} \overline{B} + \overline{B} \overline{E} + \overline{C} E + D\overline{E} + \overline{A} \]

\[ = \overline{C} + E + \overline{A} \overline{B} + \overline{A} + \overline{B} \overline{E} + D\overline{E} \]

\[ = \overline{C} + \overline{B} + \overline{A} + E \]

\[ = \overline{A} + \overline{B} + C + E \]
2.) Produce the minimum Sum of Products equation for the following K-Map.

\[ f = BC + \overline{B} \overline{D} \]
3.) For the following circuit, fill in the timing diagram. Two copies of the timing diagram are provided in case you make a mistake – circle your answer. The vertical lines are spaced one gate delay apart. Assume all gates have 1 gate delay.

Timing Diagram:

Spare (Only need to do one):
4.) Our crack design team put together the following FSM. However, they goofed. There are two obvious errors in this state machine. Please clearly identify both of them.

Error 1: **State A:** what if $x=0$ & $y=1$. This matches two edges.

Error 2: **State C:** No edge for $\overline{x} \overline{y} \overline{z}$
5.) 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. Your state diagram should be as simple as possible.

With the rush-rush of modern times, we need to slow things down. Specifically, I want a circuit that takes in one input S, and produced an output V. V will always be equal to whatever was input on S two cycles ago. For the first two cycles V will always be 0’s, since there is no “two cycles ago” for the first two cycles. Thus, if we get the input sequence “1 1 0 1 1 1 1…”, we would output “0 0 1 1 0 1 1…”
6.) In class we gave the following Verilog code, though I have added the state assignment. Implement this circuit, using premade DFFs and any basic gates (inverters, AND, OR, NAND, NOR). Your circuit should be as simple as possible.

```verilog
module simple (clk, reset, w, out);
    input  logic clk, reset, w;
    output logic out;

    logic [1:0] ps, ns;

    parameter [1:0] A = 2'b00, B = 2'b01, C = 2'b11;

    always_comb begin
        case (ps)
            A: if (w)    ns = B;
                else    ns = A;
            B: if (w)    ns = C;
                else    ns = A;
            C: if (w)    ns = C;
                else    ns = A;
            default:    ns = 2'bxx;
        endcase
    end

    assign out = (ps == C);

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

Table:

<table>
<thead>
<tr>
<th>ps</th>
<th>p50</th>
<th>w</th>
<th>out</th>
<th>n5</th>
<th>n50</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Out = p51

n50 = w

n51 = p50 * w
Out = P5,  \( N5_i = P5_0 \times W \)  \( N5_o = W \)

Name: ___________________

(Extra space to answer previous question):

\[ \text{Reset} \Rightarrow 00 \]