1) For the function $F$ given by this truth table:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

a. Express $F$ as a sum of standard products.
b. Simplify the SOP expression by combining product terms.
c. Express $F'$, the complement of $F$, as a sum of standard products. Then apply DeMorgan’s law to find the expression of $F$ as a product of standard sums.
d. Simplify the POS expression by combining sum terms.

2) Simplify these functions using K-Maps to form a product of sums:

a. $F(A, B, C, D) = \Sigma(3, 4, 5, 6, 7, 10, 11, 14, 15)$
b. $F(w, x, y, z) = \Sigma(0, 1, 2, 8, 9, 10, 12, 13)$

3) Simplify these functions using K-Maps to form a sum of products:

a. $F(w, x, y, z) = \Sigma(0, 1, 2, 8, 9, 10, 12, 13)$
b. $F(a, b, c, d) = \Sigma(0, 1, 2, 3, 4, 6, 8, 9)$

4) In class, we looked at the truth table for one segment of a 7-segment digit display. Now let’s look at a different segment. Consider the Boolean function $F(b_3, b_2, b_1, b_0)$ that implements the functionality of the bottom-left vertical segment of the display (the part that turns on for the digits 0, 2, 6, and 8). The inputs to the function are the four bits of a binary-coded decimal digit, with $b_3$ corresponding to the MSB.

a. Write out the truth table for the function, and remember to indicate which function outputs we don’t care about.

b. Use a K-Map to find a simplified expression for the function. Remember that you can use the don’t-care outputs as either zeros or ones – whichever makes the final function simpler.
5) In computer graphics programming, a *bitmap* is an image made of 0's and 1's. 8x8 pixel bitmaps can be represented as a list of 8 two-digit hexadecimal numbers (because each hex digit corresponds to exactly four bits). On an 8x8 grid, draw the bitmap encoded by these numbers by coloring in the squares corresponding to the binary 1's:

3C, 7E, DB, DB, FF, BD, 42, 3C

The first number should correspond to the top row of the image, and the LSB should be in the rightmost column.

6) Add these binary numbers in eight-bit 2's complement arithmetic. Show your work. If any of the computations result in an overflow, indicate so in your answers. When you’re finished, convert back all the numbers to decimal.

   a.  00010010 + 00010101
   b.  10001011 + 11011000

7) Convert these decimal numbers to eight-bit 2's complement representations, and then compute the addition in binary. Show your work. If any of the computations result in an overflow, indicate so in your answers. When you’re finished, convert your answers back to decimal.

   a.  112 + 17
   b.  35 + (-24)

8) Go through the Quine-McCluskey algorithm for the function

\[ F(w, x, y, z) = \Sigma(0, 1, 4, 5, 9, 13, 14, 15) \]

Use the worksheets from class if you need a reminder on how the algorithm works.

   a.  Write out the minterms of \( F \), and combine all possible four-variable terms into three-variable terms, all possible three-variable terms into two-variable terms, and so on, until no further combinations are possible.

   b.  Starting from the product terms left over after combining all possible terms, construct a prime implicant table to find a minimal representation of \( F \).