Problem 1: Write a Verilog description of a signed adder that adds a 30-bit signed quantity to a 10-bit signed quantity, producing a 31-bit result. The Verilog description can use the add operator but cannot use integers or reals. Other than declarations, a correct solution includes one line.

module prob1(sum,a,b);
    input [29:0] a;
    input [9:0] b;
    output [30:0] sum;

    // Sign extend the operands to the same size as the sum.
    assign sum = {a[29],a} + { b[9] ? 2'h1ffff : 2'h0, b };
endmodule

Problem 2: Show the longhand steps for dividing 101010 by 111. Identify the quotient and remainder.

```
  101010 |  
- 111   |  
---------| 0  
  101010 |  
- 11100 |  
---------| 01 
    1110 |  
- 1110 |  
---------| 011
      0 |  
- 111 |  
--------| 0110 <- Quotient
      0 <------------ Remainder
```
**Problem 3:** In class both levels of a two-level CLA were themselves CLAs.

(a) Write a Verilog description of a 40-bit two-level CLA in which both levels are CLAs. The design should use 10-bit CLAs for the first level; assume they are already designed. (The description should instantiate four 10-bit CLAs. Make up a name for them.)

```verilog
module cla40(cout, sum, a, b);
    input [39:0] a, b;
    output [39:0] sum;
    output cout;

    // G[0] intentionally left unconnected.

    assign cin[0] = 0;
    assign cin[1] = G[0];

    cla10 c0(G[0], P[0], sum[9:0], a[9:0], b[9:0], cin[0]);
    cla10 c1(G[1], P[1], sum[19:10], a[19:10], b[19:10], cin[1]);
    cla10 c2(G[2], P[2], sum[29:20], a[29:20], b[29:20], cin[2]);
    cla10 c3(G[3], P[3], sum[39:30], a[39:30], b[39:30], cin[3]);
endmodule
```

(b) In gate delays, how fast is the two-level adder.

Ten cycles. (See the two-level CLA in [http://www.ece.lsu.edu/ee3755/2001f/105.html](http://www.ece.lsu.edu/ee3755/2001f/105.html))

(c) Suppose the 10-bit CLAs were really ripple adders. How much cheaper and how much slower would the 40-bit adder be?

The carry logic within the 10-bit adders would not be needed.

The carry in would be available at 5 cycles. The ten-bit ripple adder would take another $2 \times 10 + 1 = 21$ cycles, for a total of 26 cycles.
**Problem 4:** Modify module `streamlined_mult` so that the multiplicand is a signed number (but keeping the multiplier an unsigned number). Take advantage of the existing structure of `streamlined_mult`, don’t just make it look like `streamlined_signed_mult`.

```
module streamlined_usmult(product,ready,multiplier,multiplicand,start,clk);
    input [15:0] multiplier, multiplicand;
    input          start, clk;
    output         product;
    output         ready;

    reg [31:0]    product;
    reg [4:0]     bit;
    wire          ready = !bit;

    initial bit = 0;

    always @( posedge clk )
        if( ready && start ) begin
            bit      = 16;
            product  = { 16'd0, multiplier };
        end else if( bit ) begin

            reg lsb;

            lsb      = product[0];
            product  = {product[31],product[31:1]};
            bit      = bit - 1;

            if( lsb )

        end
endmodule
```
Problem 5: Show a radix-8 Booth table (in the same format as the radix-4 and -2 Booth tables in Set 7).

Radix-8 Booth Table

<table>
<thead>
<tr>
<th>MB</th>
<th>C</th>
<th>x</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>000</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>-4</td>
<td>8</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>-3</td>
<td>8</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>-3</td>
<td>8</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>-2</td>
<td>8</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>-2</td>
<td>8</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Problem 6: Remember the last time you bought lunch and it did not cost an even dollar amount. If you can’t assume it cost $5.12.

(a) Convert the cost to binary. Plain, old binary, not IEEE 754. From the class account issue command "dtob 5.12" to see the answer.

\[ 5.12_{10} = 101.000111101011000001010001_2 \]

(b) Convert the cost of lunch to binary scientific notation.

\[ 1.01000111101011000001010001_2 \times 2^2 \]

(c) Convert the cost of lunch to an IEEE 754 single. From the class account issue command "fp 5.12" to see the answer.

\[ 5.12_{10} = 40a3d70a_{16} \]