CSE 369 QUIZ 3

Name: _____

Student ID Number: _____

Please do not turn the page until 11:40.

Instructions

- This quiz contains 4 pages, including this cover page.
- Show scratch work for partial credit, but put your final answers in the boxes and blanks provided.
- The quiz is closed book and closed notes.
- Please silence and put away all cell phones and other mobile or noise-making devices.
- Remove all hats, headphones, and watches.
- You have 40 minutes to complete this quiz.

Advice

- Read questions carefully before starting. Read *all* questions first and start where you feel the most confident to maximize the use of your time.
- There may be partial credit for incomplete answers; please show your work.
- Relax. You are here to learn.

Question	Points	Score
(1) Building Blocks	12	
(2) Shift Registers	10	
(3) Sequential Computation	10	
Total:	**32**	

Question 1: Building Blocks [12 pts]

We want to build a **binary-to-braille decoder circuit**. Braille is represented by 6 dots, as shown below. Circles that are black/white represent LEDs that are on (1)/off (0), respectively. The binary-coded digit is given in bits N_3–N_0.
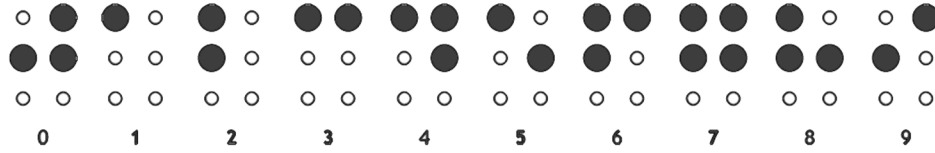
Implement the *simplest* two-level logic for an *enabled* circuit below for the **top-left dot (TL)**. You may use any 1- to 3-input logic gates discussed in the class.
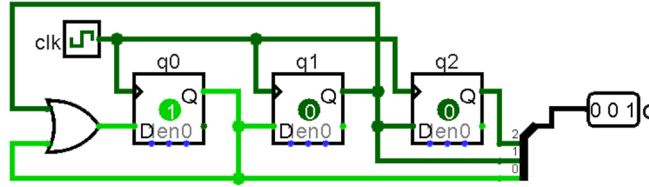
- The dot should always be off (0) if `Enable` $= 0$
- Your truth table *will be graded*.

N_3	N_2	N_1	N_0	TL
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

2

Question 2: Shift Registers [10 pts]

We are using a 3-bit LFSR as a pseudo-random number generator by connecting q1 and q0 to a 2-input OR gate. As shown below (look at Q), we are shifting bits to the LEFT.

(A) Draw out the full state transition diagram (*i.e.*, include ALL states) for this LFSR below: [4 pt]

(B) What are the "sink" state(s) of this LFSR? [1 pt]

Sink(s): ☐

(C) Complete the Verilog implementation below. [3 pt]

```verilog
module LFSR (Q, enable, reset, clk);
  input  logic enable, reset, clk;
  output logic [2:0] Q;

  always_ff @(posedge clk)
    if ( reset )          // choose a state that yields the

      Q <= 3'b_____;   //  longest non-repeating chain
    else if ( enable )

      Q <= { _____, _____, _____ };

endmodule
```

(D) Is using OR in an LFSR a good choice? *Briefly* explain. [2 pt]

3

Question 3: Sequential Computation [10 pts]

Implement a circuit that computes the **Fibonacci sequence $F_n = F_{n-1} + F_{n-2}$**. Note that it will take about n clock cycles to compute F_n and we will let it run infinitely (*no stop condition*).

- Both registers (after a `Reset`, which is not shown) start with value 0, but you will need to make sure that your circuit doesn't get stuck there! It is suggested that you tackle this part *last*. <u>Hint</u>: you need to detect this very special situation (both registers with value 0).

- You can freely use gates and routing elements discussed in class plus the constants `0` and `1` and the following logic blocks (where `w` is the bus width of our Fibonacci circuit):