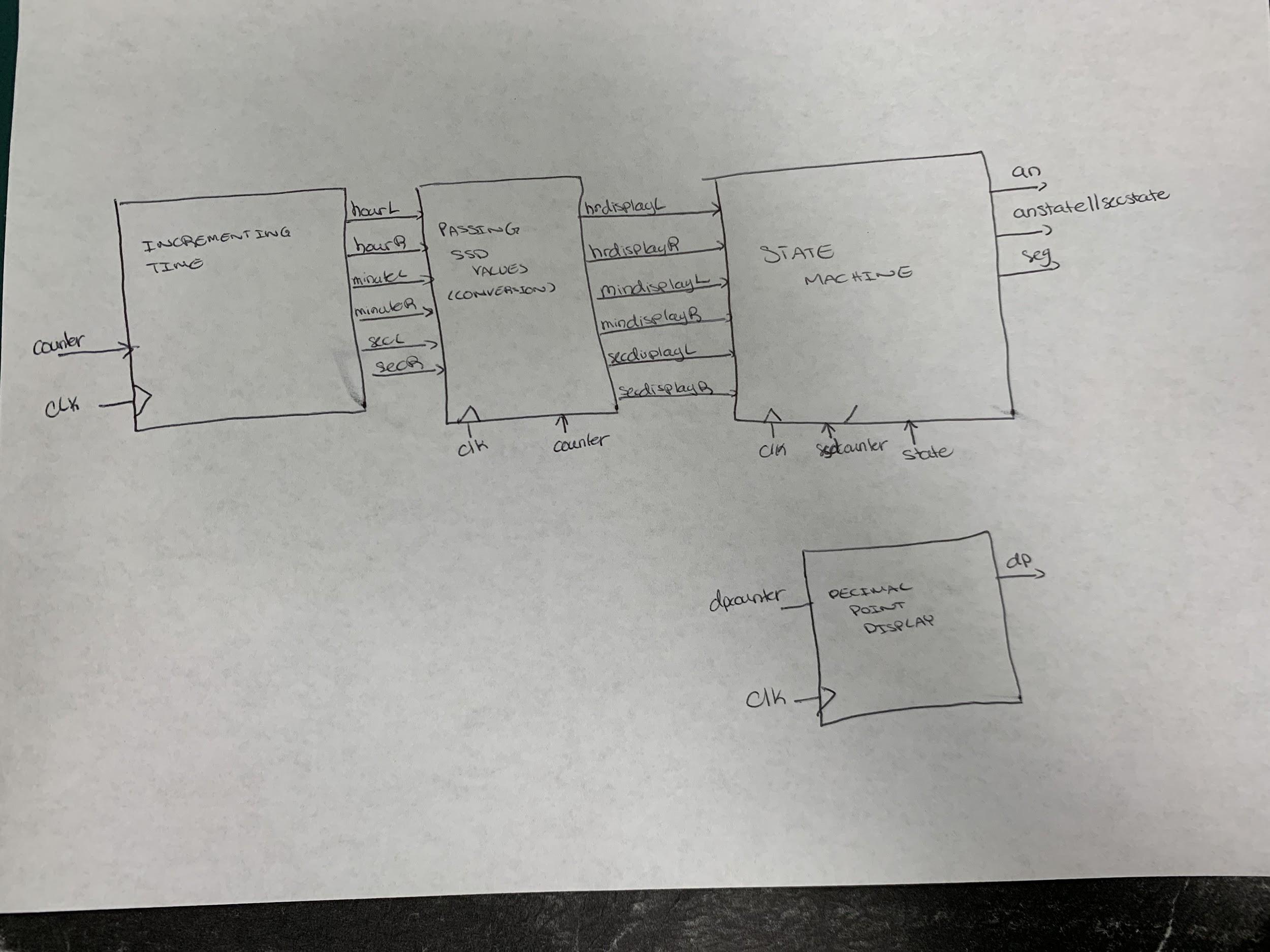
Brandon Fong

819295224

CompE 470L

**DIGITAL CLOCK LAB REPORT**

In my source code I have five major always blocks that communicate with each other. I have two blocks that controls the state of the clock module. I use two states: “clock” and “seconds.” The Clock state is for the general function to display the time on the seven-segment display (SSD). The Seconds state displays the seconds on the SSD. The next block is used to blink the decimal point on the SSD. Another block is to perform the incrementation for the clock. And lastly, I have one block that converts the binary values of the time to the SSD values to display a readable time.



The State Machine always block controls what is displayed to the user. The block uses a counter (“ssdcounter”) to count to the value 500,000. That value multiplexes the digits. Each digit is on until the counter reaches this value then it switches states to turn on a different segment. As you can see, I have nested a case statement in another case statement. The purpose is to perform the extra credit to display the seconds on the SSD. To change the state to display seconds, I use another always block to detect whether switch number 1 is switched to its on state and its performance is identical to that of the previously mentioned.

The simple out of the group is the decimal point display. I use counter to count to half a second then I toggle the decimal point. I use an if-else statement to detected if the user is attempting to change/set time. When the user is setting the time, the decimal pointer will stop blink and will resume when the user sets the time.

The next block is the incrementation. This was the most difficult because of the synchronous property. I have three major counters: counter, setcounter, and seccounter. At the positive edge of the clock, I increment the time everytime counter counts to one second. Initially I started with just incrementing the minutes then when I added the seconds I made a second counter “seccounter.” Each segment on the SSD is named hourL, hourR, minuteL, and minuteL (from left to right respectively). Starting from the right, if minuteL = 9 then increment minuteR and set minuteL to 0. When minuteR = 5, increment hourR and set minuteR to 0. When hourR == 9, increment hourL and set hourR to 0. If both hourL = 1 and hourR = 2, then set hourL to 0 and hourR to 1. Same goes for the seconds registers. I have an initial block in the beginning to set the time to 12:00am when the program starts. This function mentioned above is in an if-else statement as the else case. I have two if statements to set time. The first detects switch 0 and resets seconds registers and counters. I made so that the buttons can set the time. Pressing the right button will increment the minutes. The left will increment the hours. The top button will turn on LED 0 and the bottom button will turn on LED 1. The second if statement allows the user to set the time with binary values. I use the 4 left most switches and detect overflow. For example, if you’re trying to set minuteR to 6, it will detect the overflow and will put 5 instead.

My last major block is passing the SSD values to make the time readable to the user. The values used in this block was based off from lab 3. I used 6 case statements to compare the minutes, hours, and seconds segment displays. This function is synchronous to the clock and will perform everytime the counter reaches one second. I don’t increment this counter in this block since it is already incrementing in another block.

**SOURCE CODE:**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 03/05/2019 03:33:58 PM

// Design Name:

// Module Name: clock

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module clock(

input clk, //100MHz

input btnC, btnU, btnL, btnR, btnD,

input [15:0] sw,

output reg [6:0] seg,

output reg dp,

output reg [3:0] an,

output reg [15:0] LED);

parameter n = (100000000/50), m = (100000000/500000), sec = 100000000;// min = (60/100000000), hr = (120/100000000);

// n = counter for 50Hz, m = counter for 5ms

parameter seg1 = 0, seg2 = 1, seg3 = 2, seg4 = 3, clock = 4, seconds = 5;

reg [31:0] ssdcounter, dpcounter, seccounter, counter, setcounter;

reg[31:0] systemstate, secstate, anstate;

//reg [31:0] HRDL, HRDR, MDL, MDR;

reg [31:0] hrdisplayL, hrdisplayR, mindisplayL, mindisplayR, secdisplayL, secdisplayR;

wire [15:0] hR, hL, mR, mL, sR, sL;

reg[15:0] hourL, hourR, minuteL, minuteR, secL, secR;

initial begin

secL = 0;

secR = 0;

minuteR = 0;

minuteL = 0;

hourR = 2;

hourL = 1;

LED[0] = 1; //AM

//LED[1] = 0; //PM

systemstate = clock;

end

//System states

always @(\*) begin

if(sw[1]) systemstate = seconds;

else systemstate = clock;

end

//SSD

always @(posedge clk) begin

if(ssdcounter == n/4) begin

ssdcounter <= 0;

case(systemstate)

clock: begin

case(anstate)

seg1: begin

an <= 4'b0111;

anstate <= seg2;

seg <= hrdisplayL;

end

seg2: begin

an <= 4'b1011;

anstate <= seg3;

seg <= hrdisplayR;

end

seg3: begin

an <= 4'b1101;

anstate <= seg4;

seg <= mindisplayL;

end

seg4: begin

an <= 4'b1110;

anstate <= seg1;

seg <= mindisplayR;

end

//default an <= 0;

endcase

end

seconds: begin

case(secstate)

seg1: begin

an <= 4'b0111;

secstate <= seg2;

seg <= secdisplayL;

end

seg2: begin

an <= 4'b1011;

secstate <= seg1;

seg <= secdisplayR;

end

endcase

end

endcase

end

else ssdcounter = ssdcounter + 1;

end

//decimal blink every second

always @(posedge clk) begin

if(sw[0] || sw[2]) begin

dpcounter <= 0;

dp <= 0;

end

else begin

if(dpcounter == (sec/2))begin

dpcounter <= 0;

dp <= ~dp;

end

else dpcounter <= dpcounter + 1;

end

end

//Incrementing time

//Might be wrong with the hours

always @(posedge clk) begin

if(sw[0])begin //set time incrementation

secR <= 0;

secL <= 0;

counter <= 0;

seccounter <= 0;

if(setcounter == sec/9) begin

setcounter <= 0;

if(btnR) begin

minuteR <= mR + 1;

if(minuteR == 9) begin

minuteR <= 0;

minuteL <= minuteL + 1;

if(minuteL == 5) minuteL <= 0;

end

end

if(btnL) begin

hourR <= hR + 1;

if((hourR == 9) && (hourL == 0)) begin

hourR <= 0;

hourL <= 1;

end

if((hourR == 2) && (hourL == 1)) begin

hourL <= 0;

hourR <= 1;

end

end

if(btnU) LED[0] <= ~LED[0]; //AM

if(btnD) LED[1] <= ~LED[1]; //PM

end

else setcounter = setcounter + 1;

end

if(sw[2])begin //set time binary

secR <= 0;

secL <= 0;

counter <= 0;

seccounter <= 0;

if(btnL)begin //minuteL

if(sw[15:12] > 5) minuteL <= 5;

else minuteL <= sw[15:12];

end

if(btnR)begin //minuteR

if(sw[15:12] > 9) minuteR <= 9;

else minuteR <= sw[15:12];

end

if(btnU)begin //hourR

if(sw[15:12] > 9) hourR <= 9;

else hourR <= sw[15:12];

end

if(btnD)begin //hourL

if(sw[15:12] > 1) hourL <= 1;

else hourL <= sw[15:12];

end

if((hourL == 1) && (hourR > 2)) hourL <= 0;

if((hourL == 0) && (hourR == 0)) hourR <= 1;

end

else begin

if(counter == sec) begin

counter <= 0;

seccounter <= seccounter + 1;

secR <= secR + 1;

if(secR == 9) begin

secR <= 0;

secL <= secL + 1;

if(secL == 5) secL <= 0;

end

if(seccounter == 59)begin

seccounter <= 0;

minuteR <= minuteR + 1;

if(minuteR == 9) begin

minuteR <= 0;

minuteL <= minuteL + 1;

if(minuteL == 5)begin

minuteL <= 0;

hourR <= hourR + 1;

if((hourR == 9) && (hourL == 0)) begin

hourR <= 0;

hourL <= hourL + 1;

end

if((hourR == 1) && (hourL == 1) && (minuteR == 9) && (minuteL == 5)) begin

//hourL <= 0;

//hourR <= 1;

LED[0] <= ~LED[0]; //AM

LED[1] <= ~LED[1]; //PM

end

if((hourR == 2) && (hourL == 1)) begin

hourL <= 0;

hourR <= 1;

end

end

end

end

end

else counter <= counter + 1;

end

end

//wires

assign hR = hourR;

assign hL = hourL;

assign mR = minuteR;

assign mL = minuteL;

assign sR = secR;

assign sL = secL;

//Passing SSD values

always @(\*) begin

//if(counter == sec) begin

case(hourL)

0: hrdisplayL = 7'b1000000;

1: hrdisplayL = 7'b1111001;

default hrdisplayL = 7'b1111001;

endcase

case(hourR)

0: hrdisplayR = 7'b1000000;

1: hrdisplayR = 7'b1111001;

2: hrdisplayR = 7'b0100100;

3: hrdisplayR = 7'b0110000;

4: hrdisplayR = 7'b0011001;

5: hrdisplayR = 7'b0010010;

6: hrdisplayR = 7'b0000010;

7: hrdisplayR = 7'b1111000;

8: hrdisplayR = 7'b0000000;

9: hrdisplayR = 7'b0011000;

default hrdisplayR = 7'b0100100;

endcase

case(minuteL)

0: mindisplayL = 7'b1000000;

1: mindisplayL = 7'b1111001;

2: mindisplayL = 7'b0100100;

3: mindisplayL = 7'b0110000;

4: mindisplayL = 7'b0011001;

5: mindisplayL = 7'b0010010;

default mindisplayL = 7'b1000000;

endcase

case(minuteR)

0: mindisplayR = 7'b1000000;

1: mindisplayR = 7'b1111001;

2: mindisplayR = 7'b0100100;

3: mindisplayR = 7'b0110000;

4: mindisplayR = 7'b0011001;

5: mindisplayR = 7'b0010010;

6: mindisplayR = 7'b0000010;

7: mindisplayR = 7'b1111000;

8: mindisplayR = 7'b0000000;

9: mindisplayR = 7'b0011000;

default mindisplayR = 7'b1000000;

endcase

case(secL)

0: secdisplayL = 7'b1000000;

1: secdisplayL = 7'b1111001;

2: secdisplayL = 7'b0100100;

3: secdisplayL = 7'b0110000;

4: secdisplayL = 7'b0011001;

5: secdisplayL = 7'b0010010;

default secdisplayL = 7'b1000000;

endcase

case(secR)

0: secdisplayR = 7'b1000000;

1: secdisplayR = 7'b1111001;

2: secdisplayR = 7'b0100100;

3: secdisplayR = 7'b0110000;

4: secdisplayR = 7'b0011001;

5: secdisplayR = 7'b0010010;

6: secdisplayR = 7'b0000010;

7: secdisplayR = 7'b1111000;

8: secdisplayR = 7'b0000000;

9: secdisplayR = 7'b0011000;

default secdisplayR = 7'b1000000;

endcase

// end

end

endmodule

***Instructions to set the clock***

To set the time on this clock, switch number 0 to the up position. Use the Right button to set minutes. Use the Left button to set the hour. Use the Up button to indicate AM and Down button to indicate PM. Put the switch in the down position to set time and run clock.

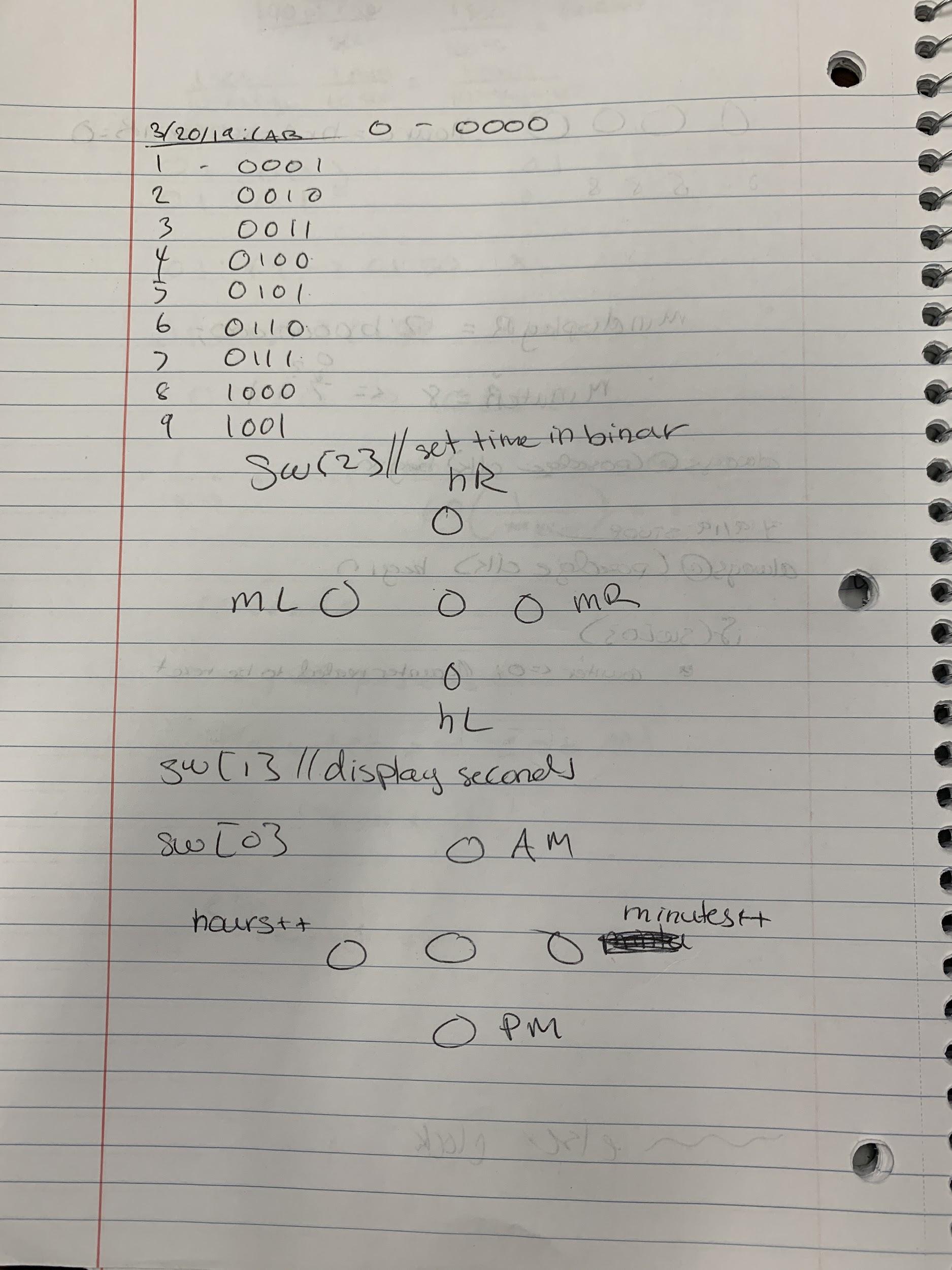
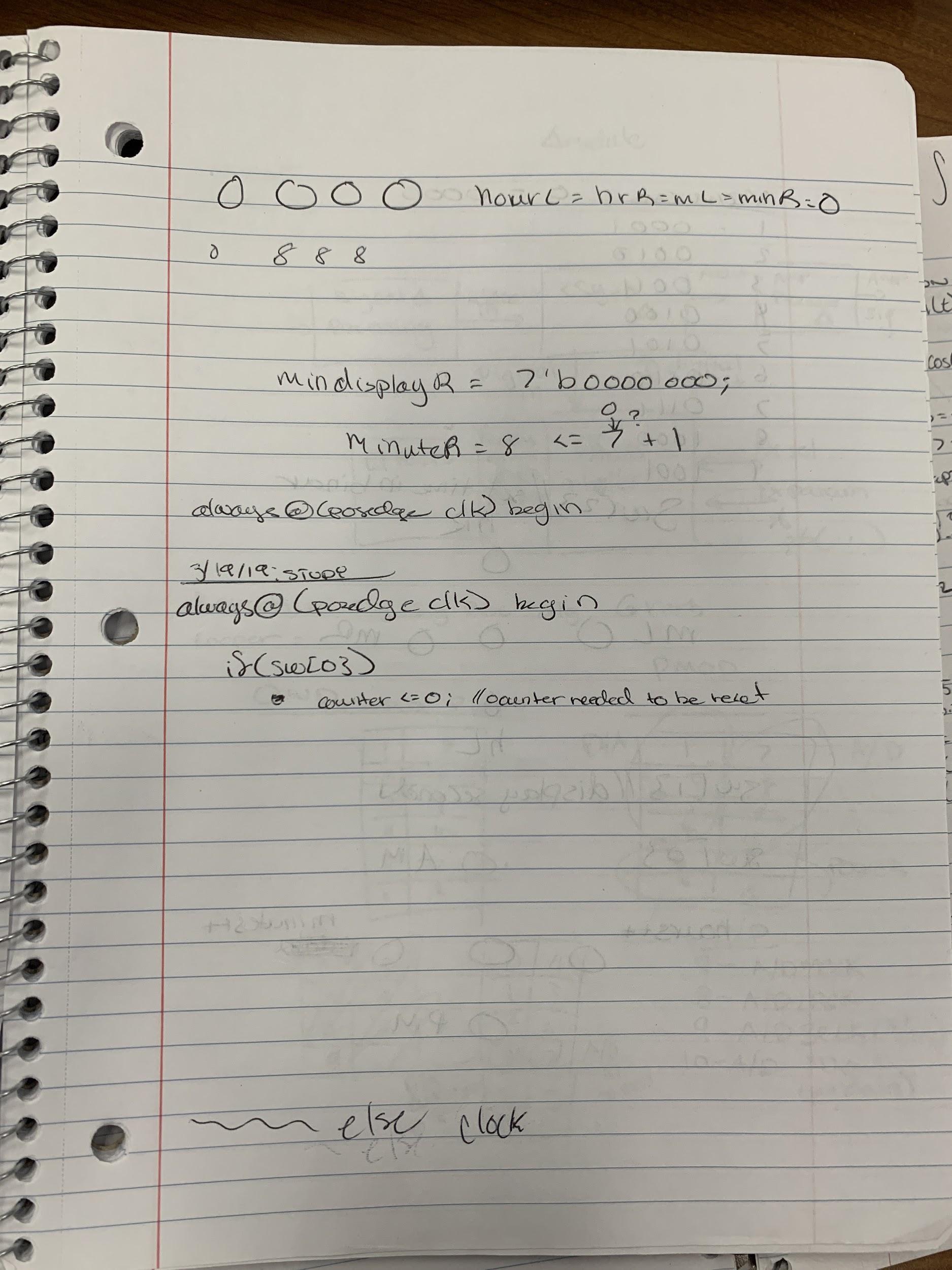
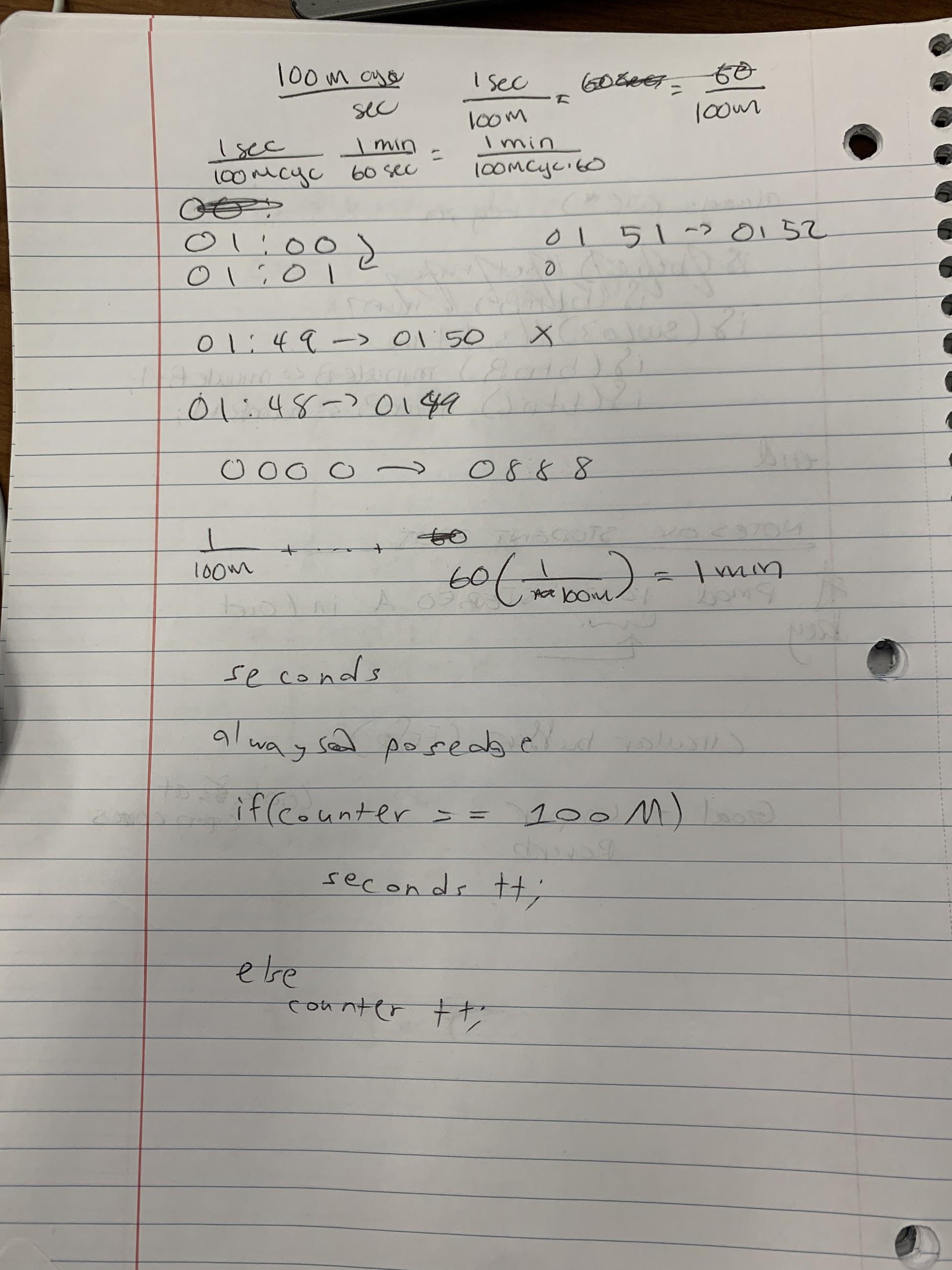
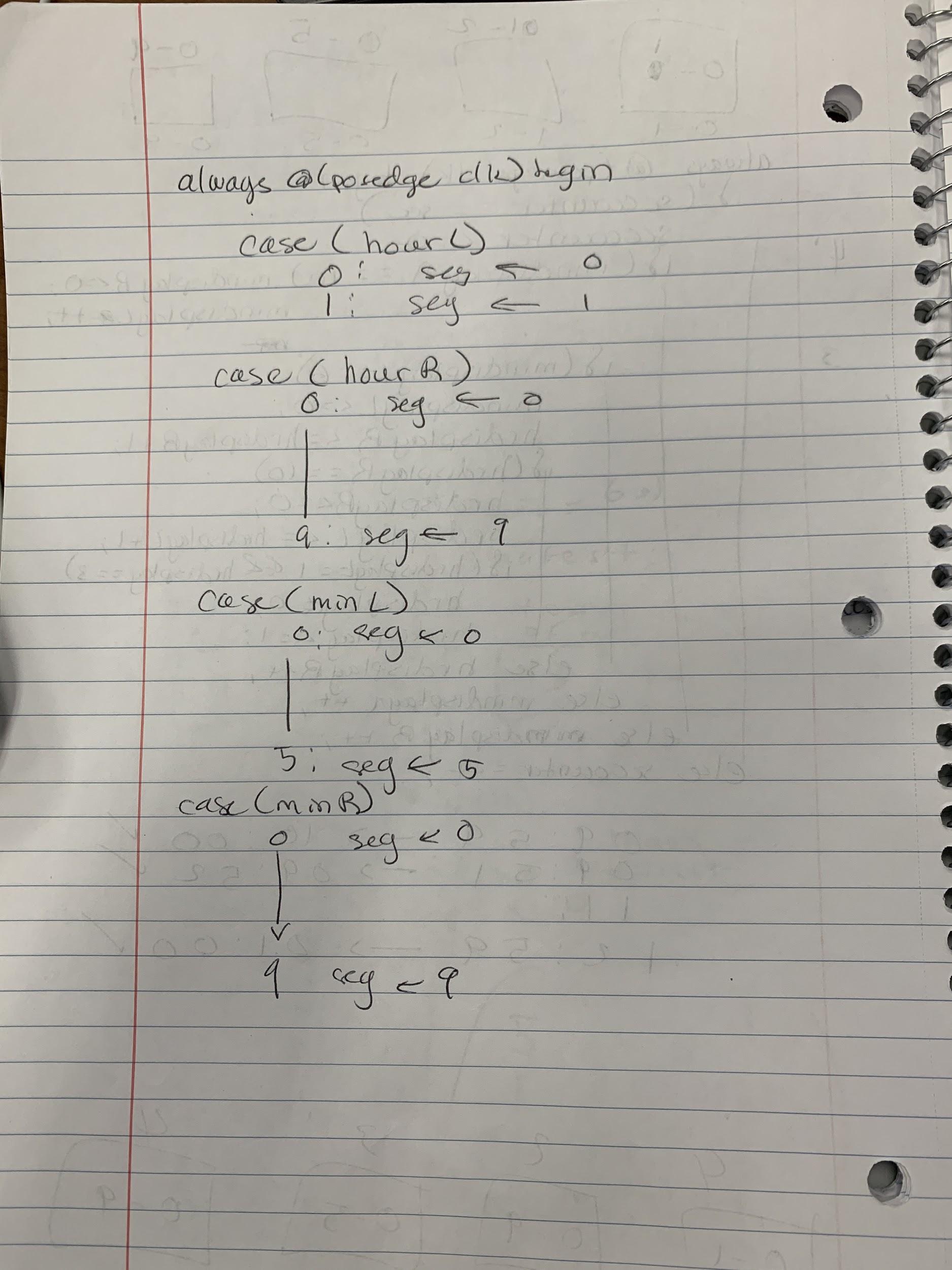
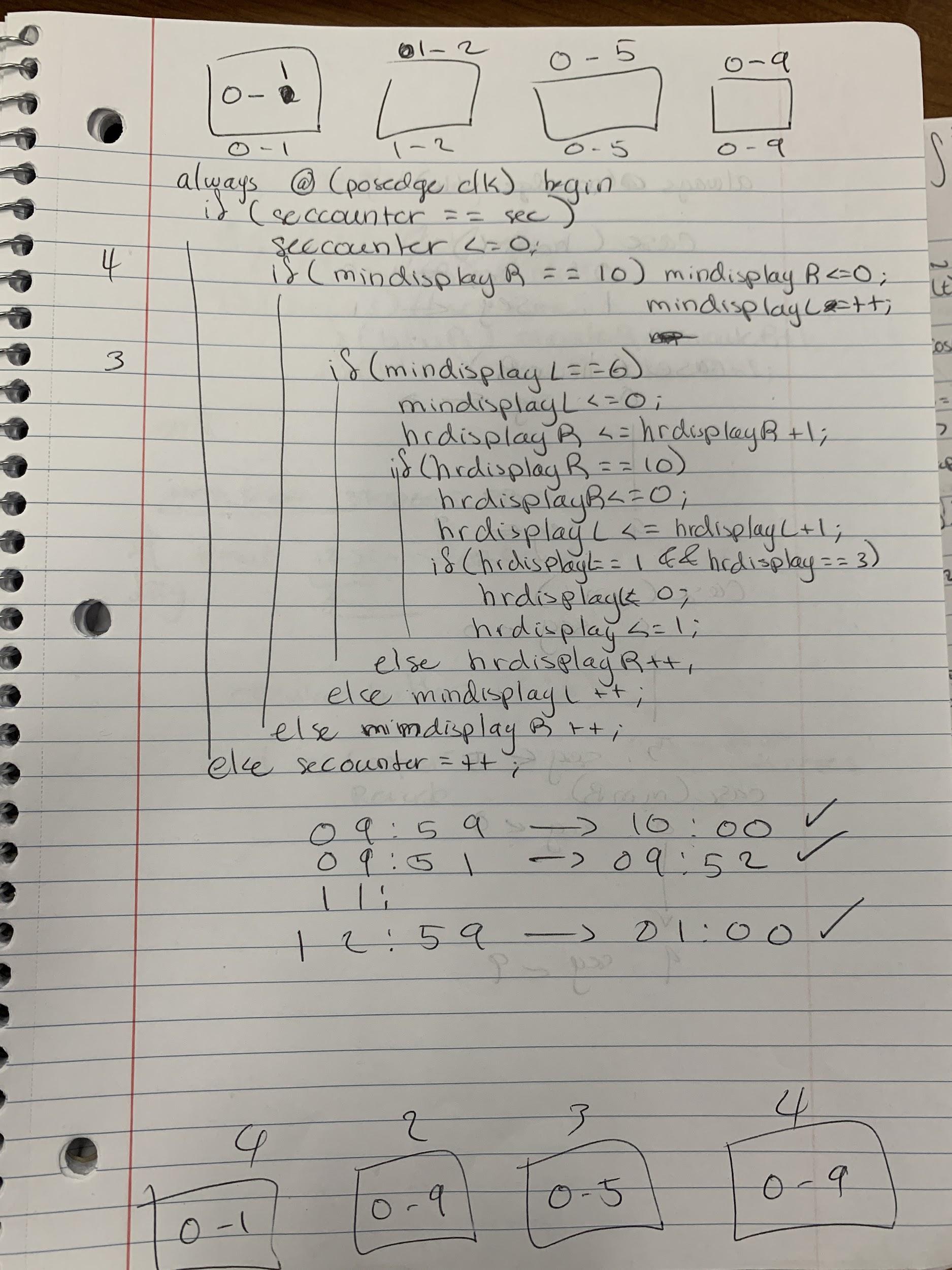
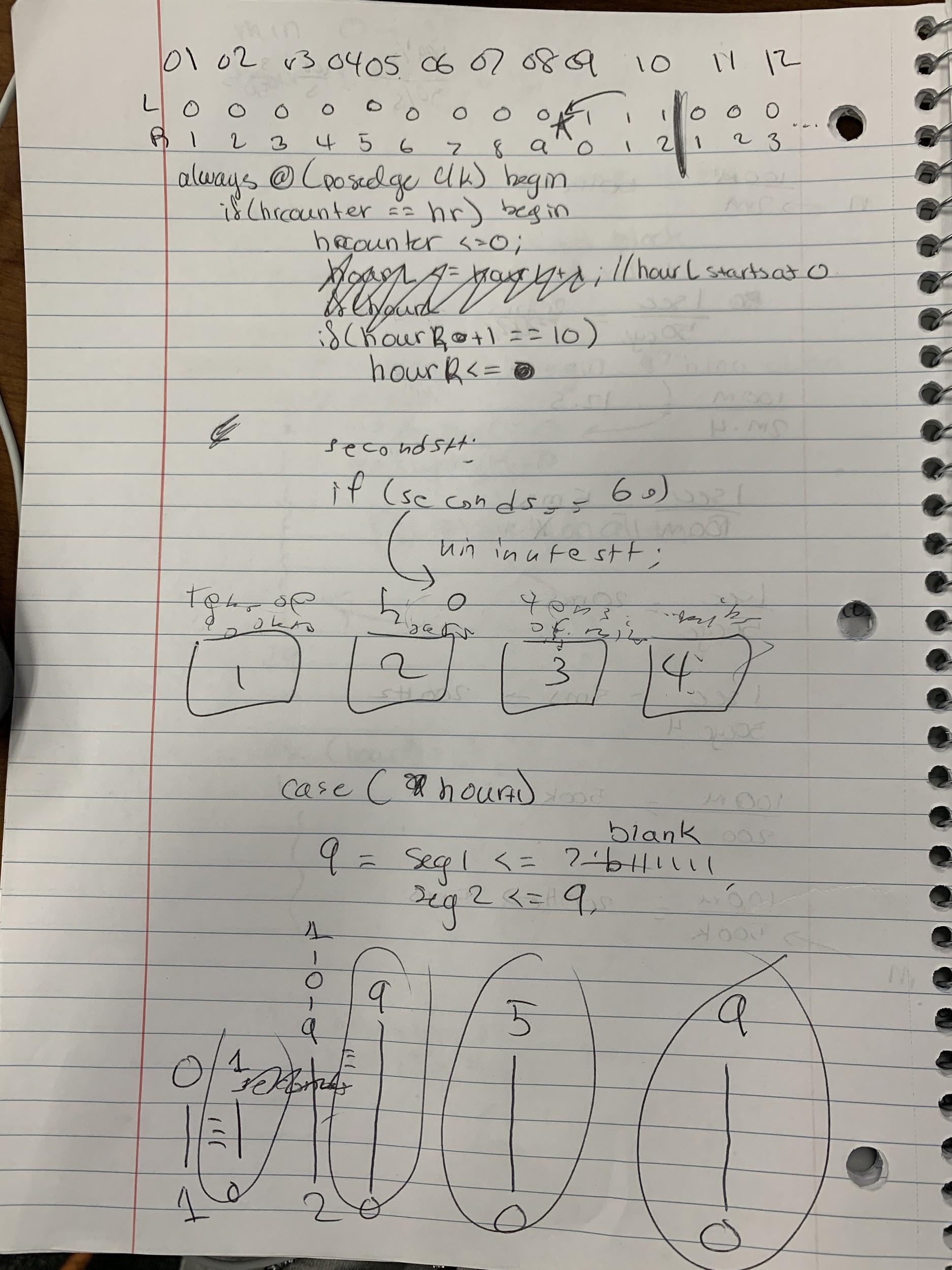
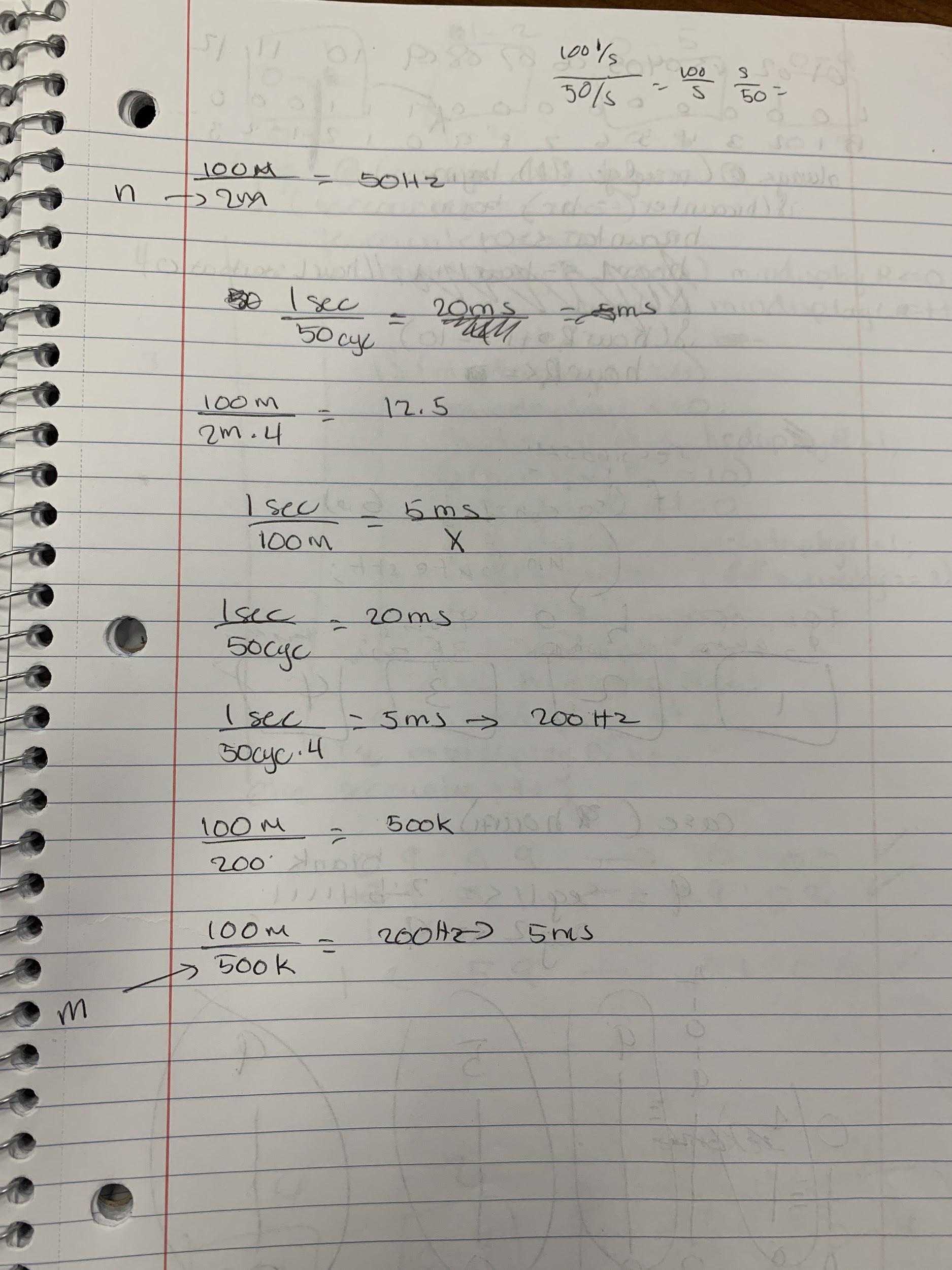
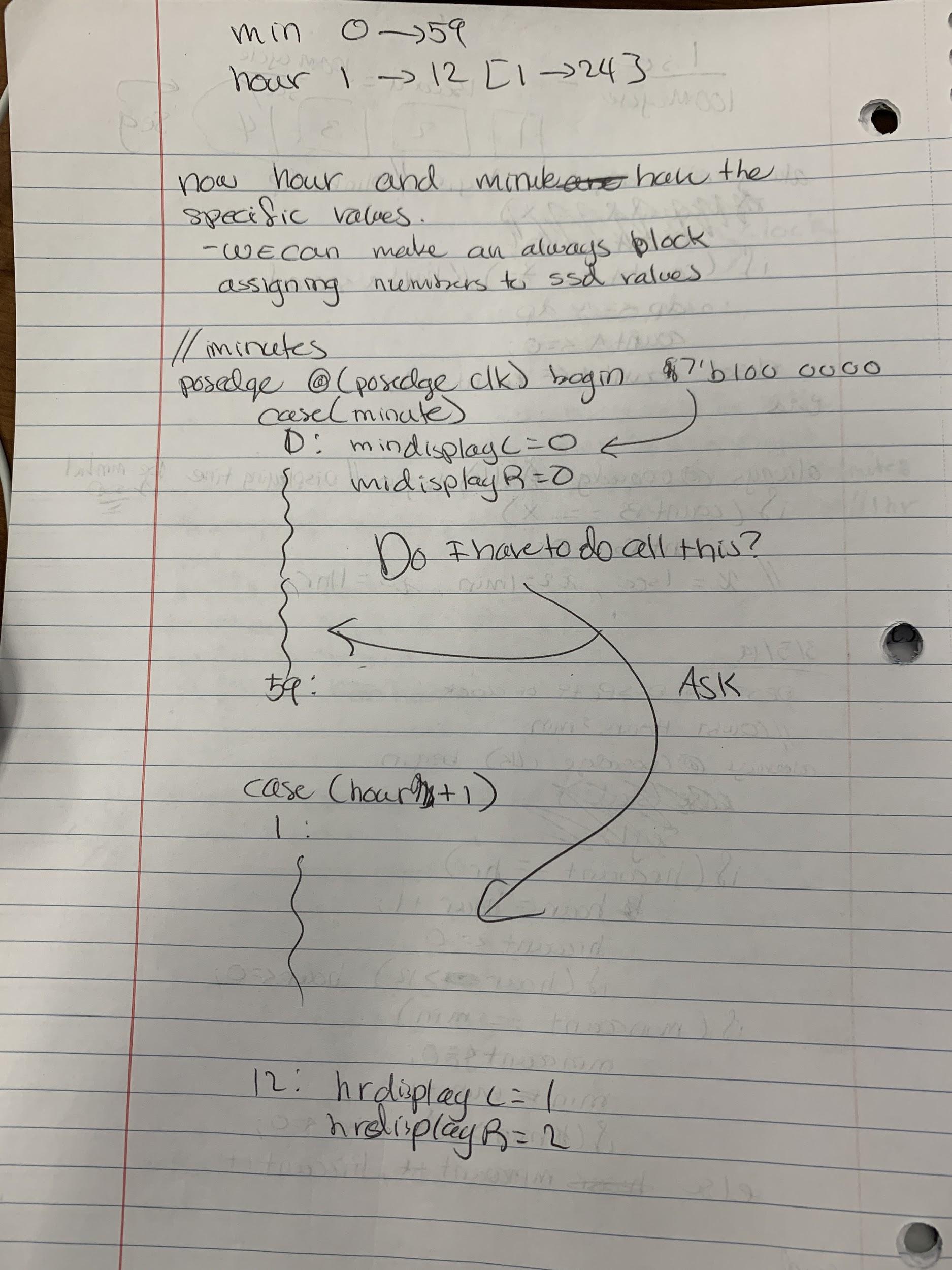
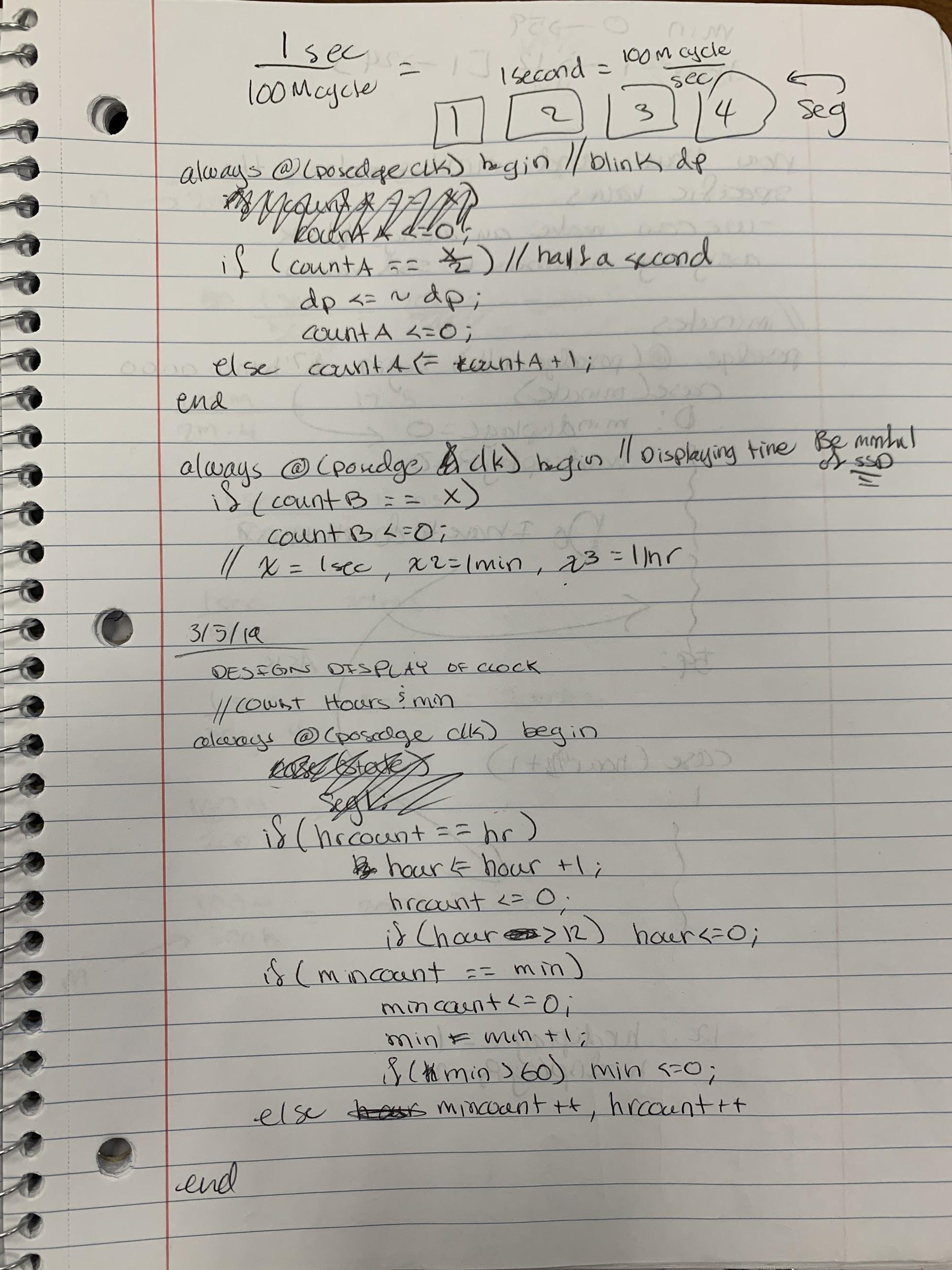
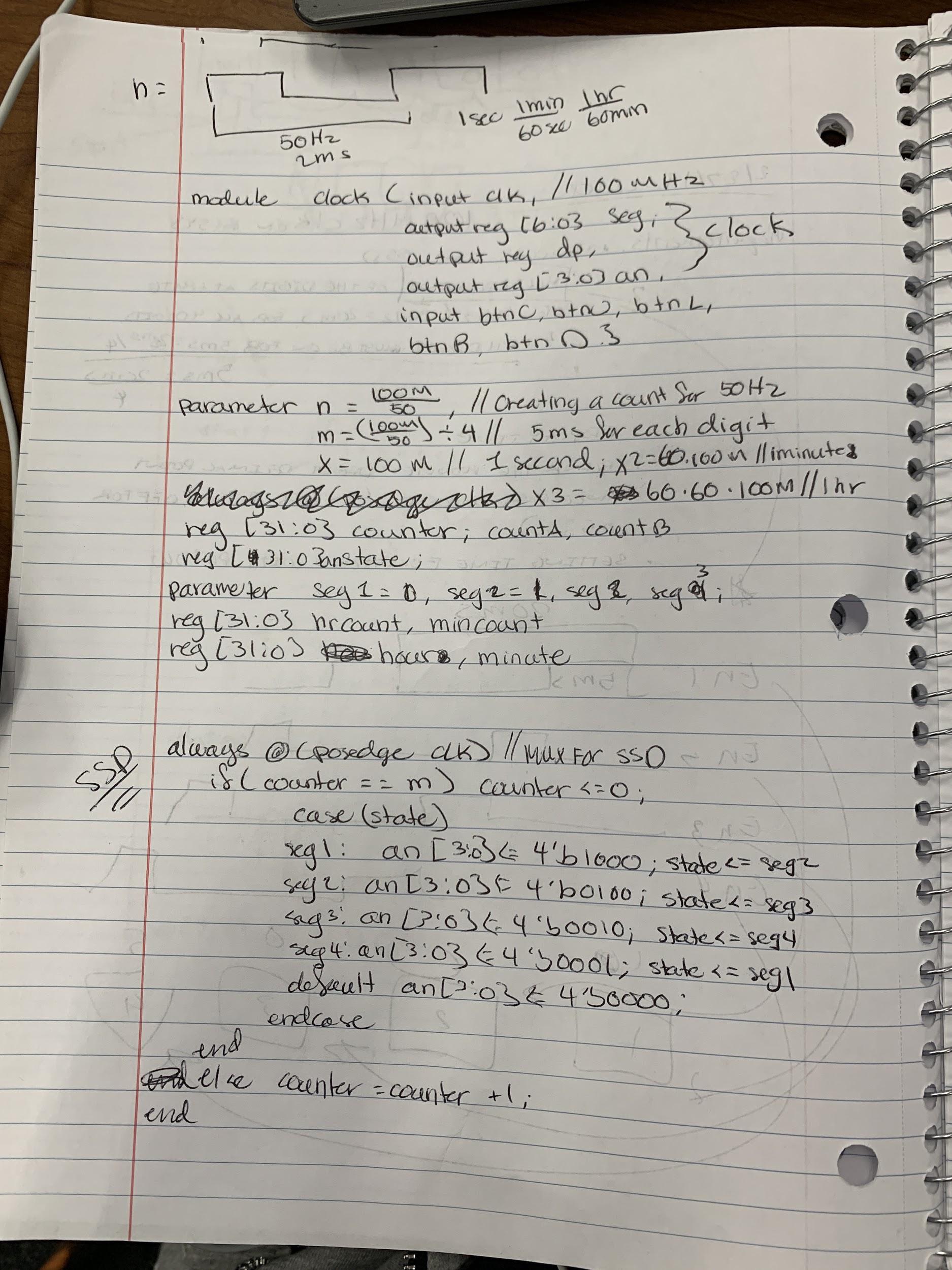
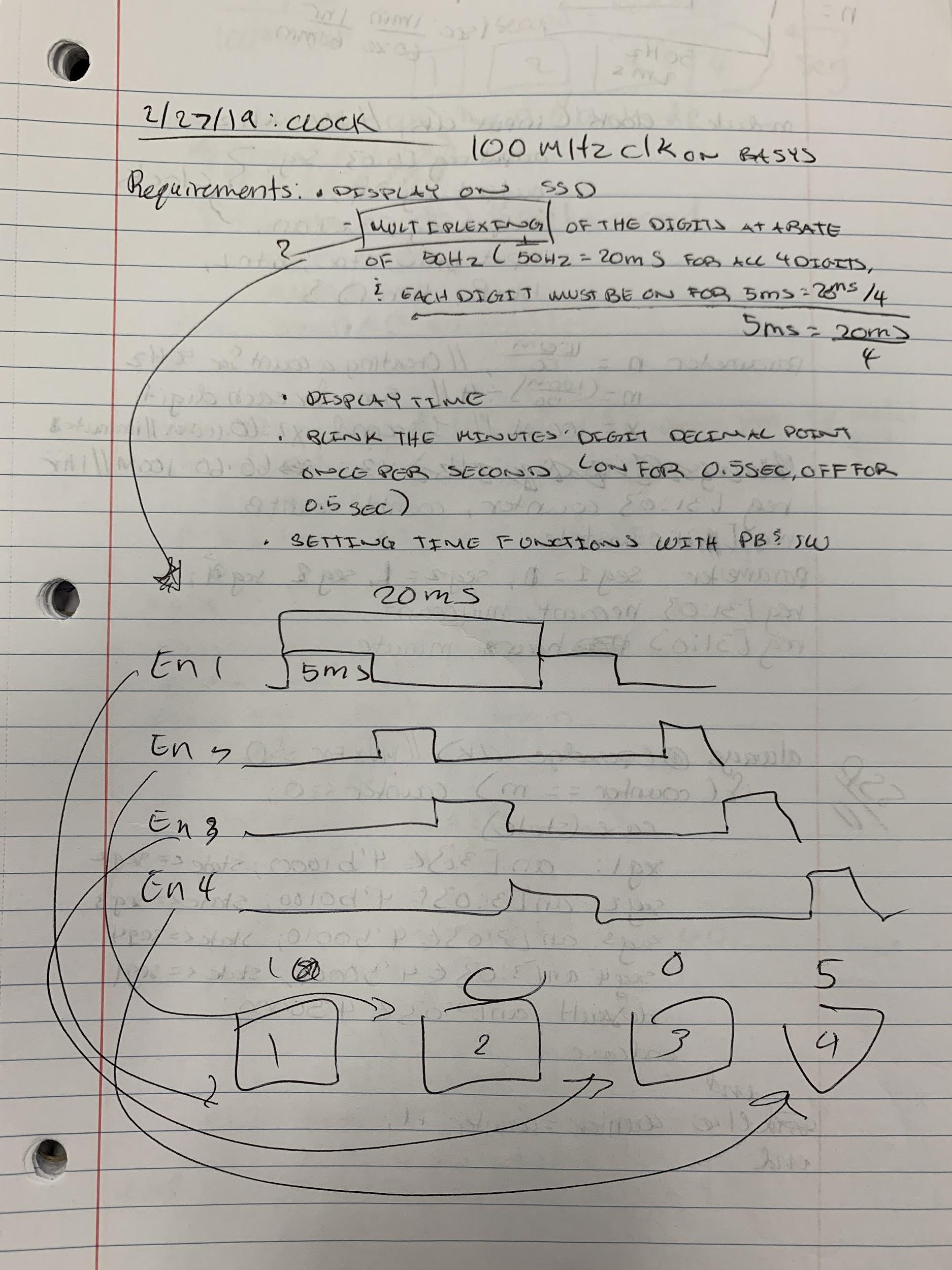
Another option for user is to pass binary values to set time. Switch number 2 to the up position and use the 4 left most switches to set binary values. Use the Left button to set the the left minute digit to the binary value. Use the Right button to set the right minute digit. Use the Up button to set the right hour digit. Use the Down button to set the left hour digit. Put the switch to the down position to the down position to run clock.

To view seconds, switch number 1 to up position.

**Test and Validation results:**

The clock works correctly, the only issue I had was initializing the LED outputs to one to represent AM/PM. I was able to toggle it when it turns from 1159 to 1200, but it toggles both. To bypass the issue, I just used the set time function to set it to either AM or PM. Other than that, everything works smoothly. Decimal point blinks every second and each digit is on for 5ms.

**Lab notes:**

****