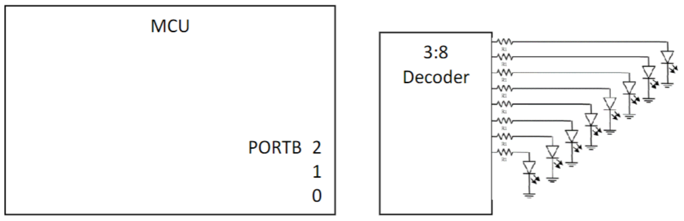
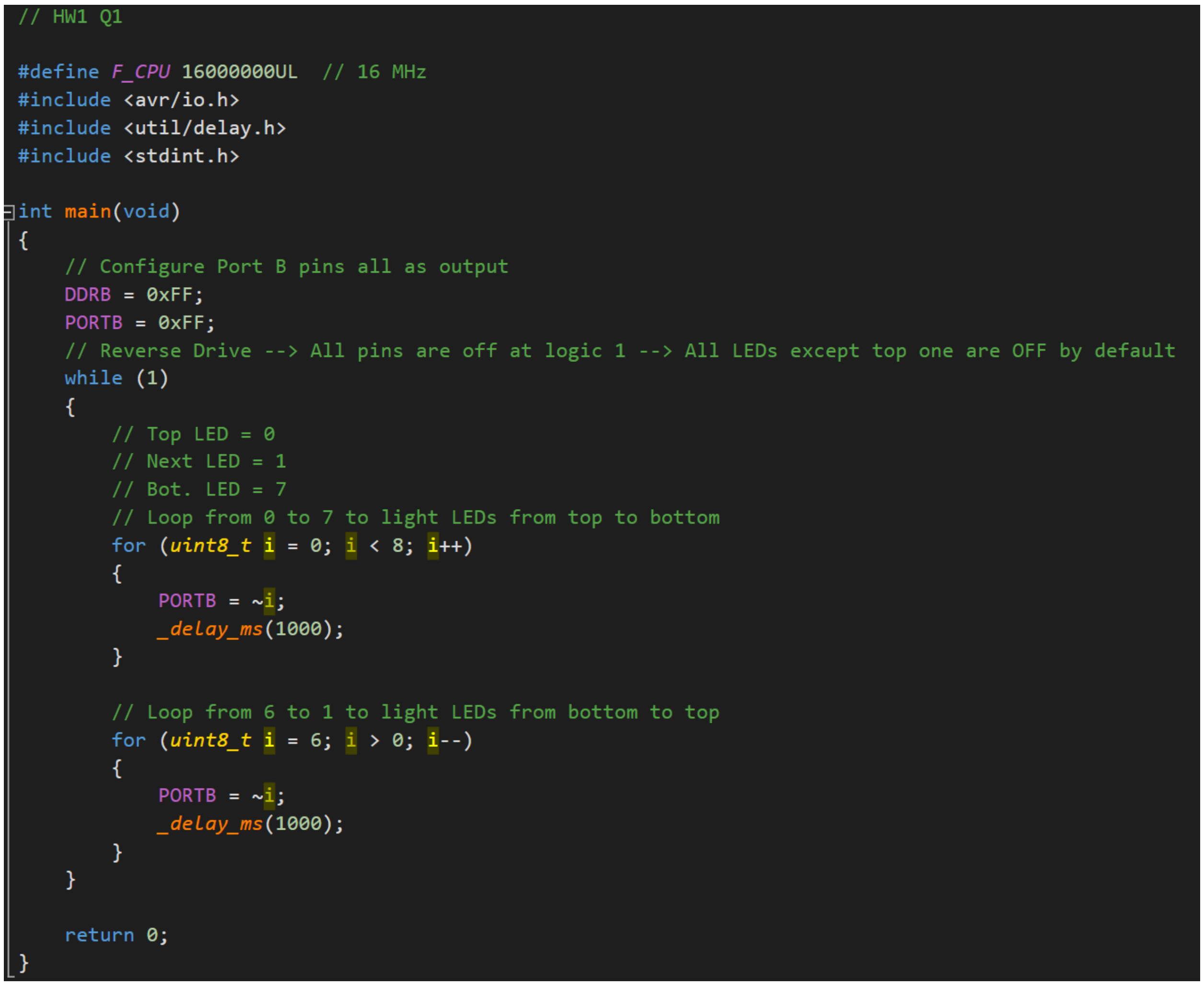
Due 9/18 – Members: Brandon V, Amy Q, Douglas G, Christian C, Brandon M

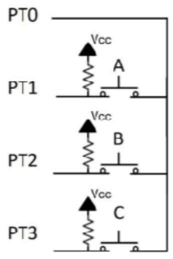
1. A 3:8 decoder converts 3 bit input into a 8 bit output. E.g.
   if input = 3 (011b), output pin 3 is HI, others are LO (00001000b);
   if input = 5 (101b), output pin 5 is HI, others are LO (00100000b).
   The decoder input is driven by port B of the microcontroller, and its output is connected to 8 LEDs in reverse-drive orientation. Write a code to make the LEDs light sequentially from top to bottom and back to top in an infinite loop.

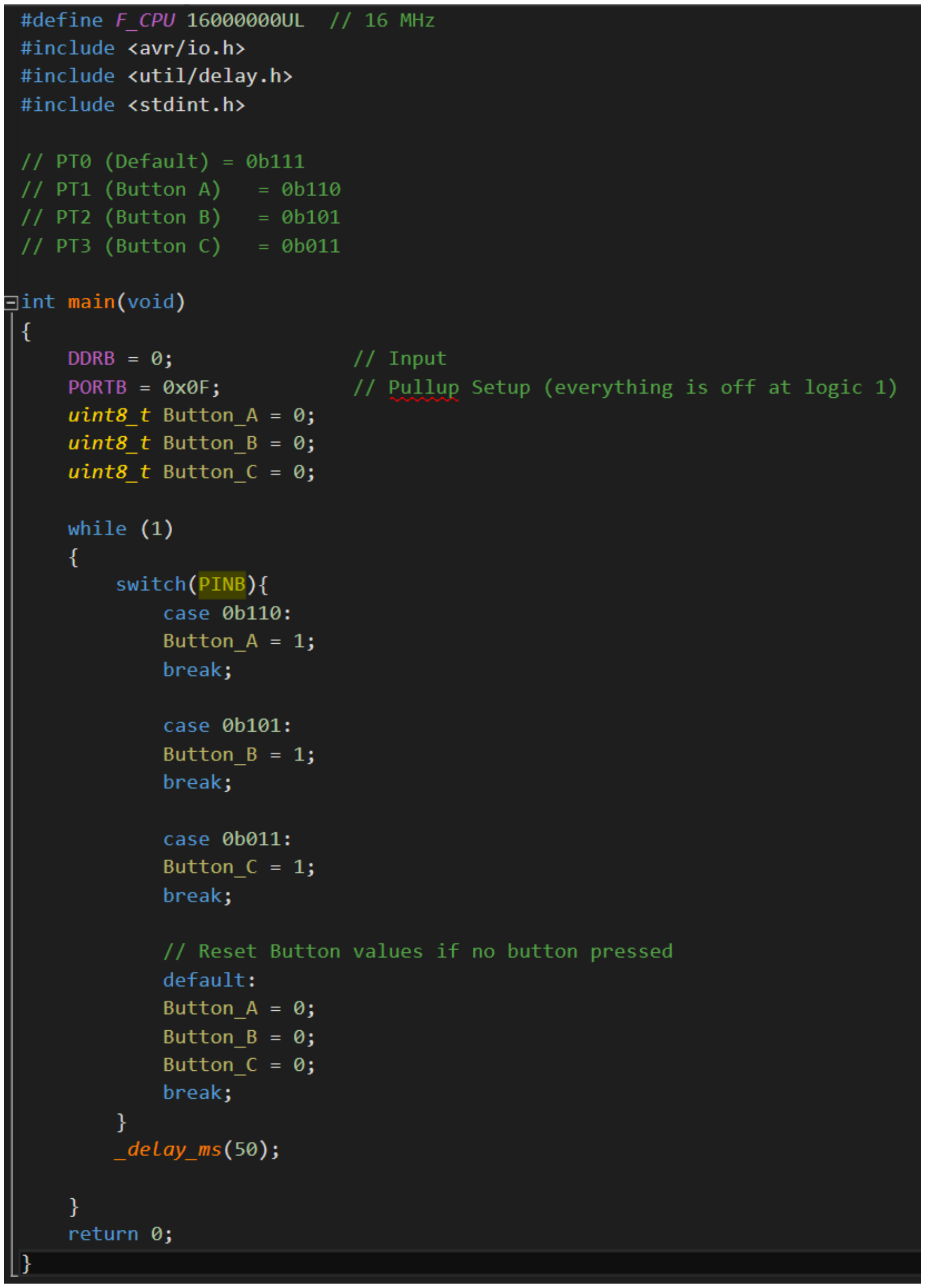




```
// HW1 Q1

#define F_CPU 16000000UL  // 16 MHz
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>

int main(void)
{
    // Configure Port B pins all as output
    DDRB = 0xFF;
    PORTB = 0xFF;
    // Reverse Drive --> All pins are off at logic 1 --> All LEDs except top one are OFF by default
    while (1)
    {
        // Top LED = 0
        // Next LED = 1
        // Bot. LED = 7
        // Loop from 0 to 7 to light LEDs from top to bottom
        for (uint8_t i = 0; i < 8; i++)
        {
            PORTB = ~i;
            _delay_ms(1000);
        }

        // Loop from 6 to 1 to light LEDs from bottom to top
        for (uint8_t i = 6; i > 0; i--)
        {
            PORTB = ~i;
            _delay_ms(1000);
        }
    }

    return 0;
}
```

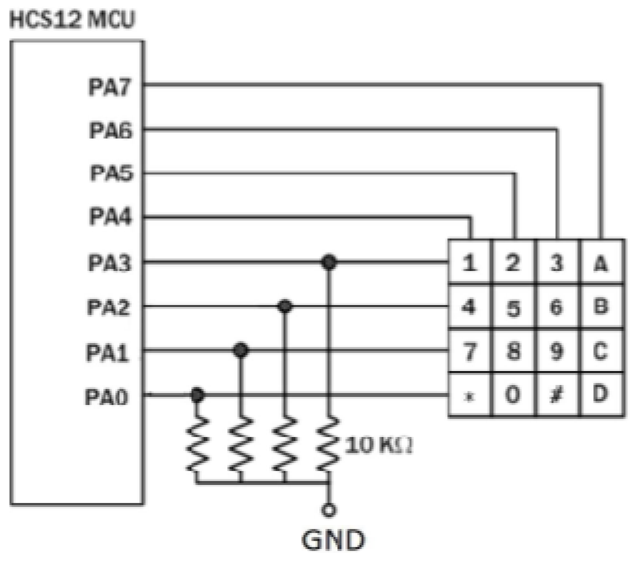
2. Given the circuit in Figure 2 (see attached file). Select a port and write a program that will determine which button is pressed.





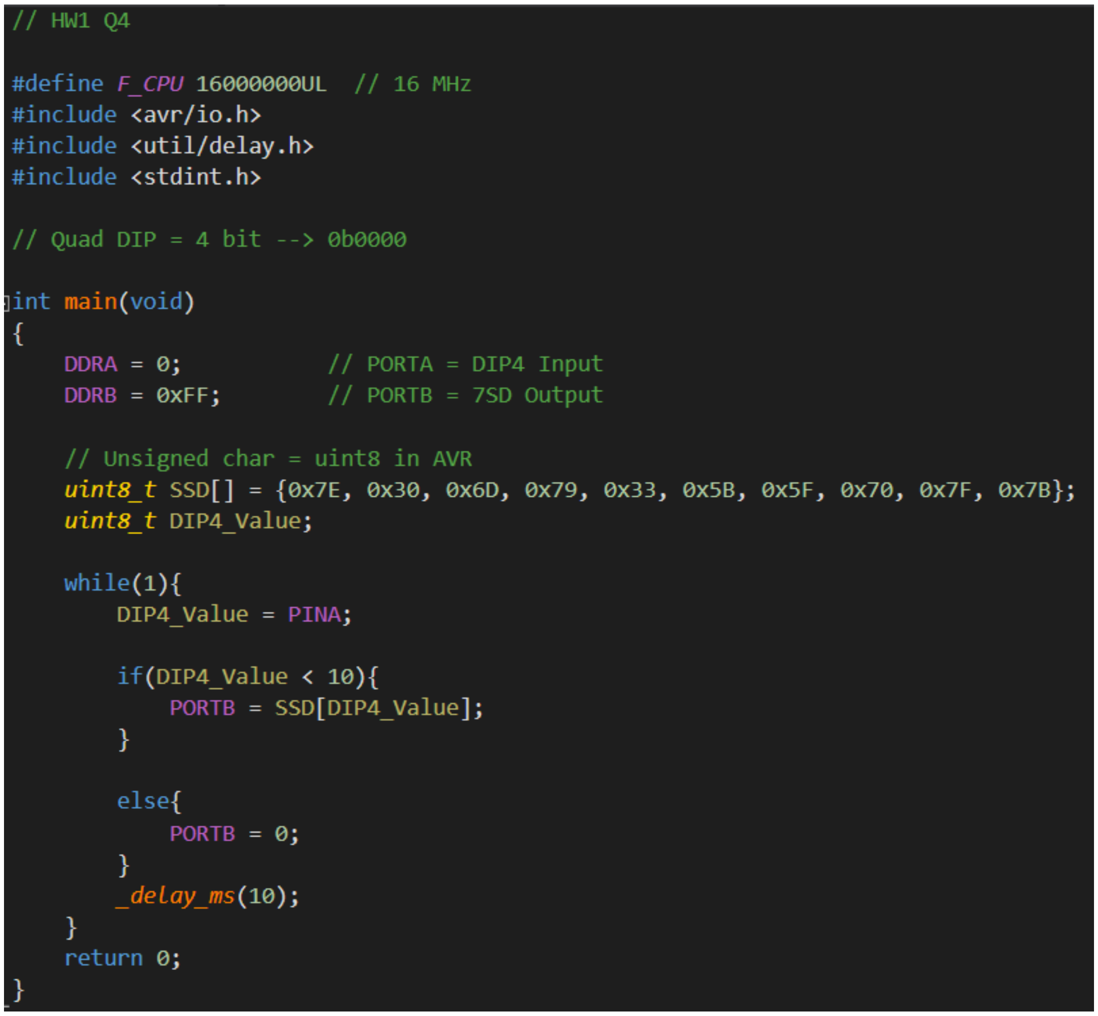
```
#define F_CPU 16000000UL  // 16 MHz
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>

// PT0 (Default) = 0b111
// PT1 (Button A)   = 0b110
// PT2 (Button B)   = 0b101
// PT3 (Button C)   = 0b011

int main(void)
{
    DDRB = 0;                // Input
    PORTB = 0x0F;            // Pullup Setup (everything is off at logic 1)
    uint8_t Button_A = 0;
    uint8_t Button_B = 0;
    uint8_t Button_C = 0;

    while (1)
    {
        switch(PINB){
            case 0b110:
            Button_A = 1;
            break;

            case 0b101:
            Button_B = 1;
            break;

            case 0b011:
            Button_C = 1;
            break;

            // Reset Button values if no button pressed
            default:
            Button_A = 0;
            Button_B = 0;
            Button_C = 0;
            break;
        }
        _delay_ms(50);

    }
    return 0;
}
```

3. For the circuit in figure 3, derive a table to determine the values in PORTA for each key.

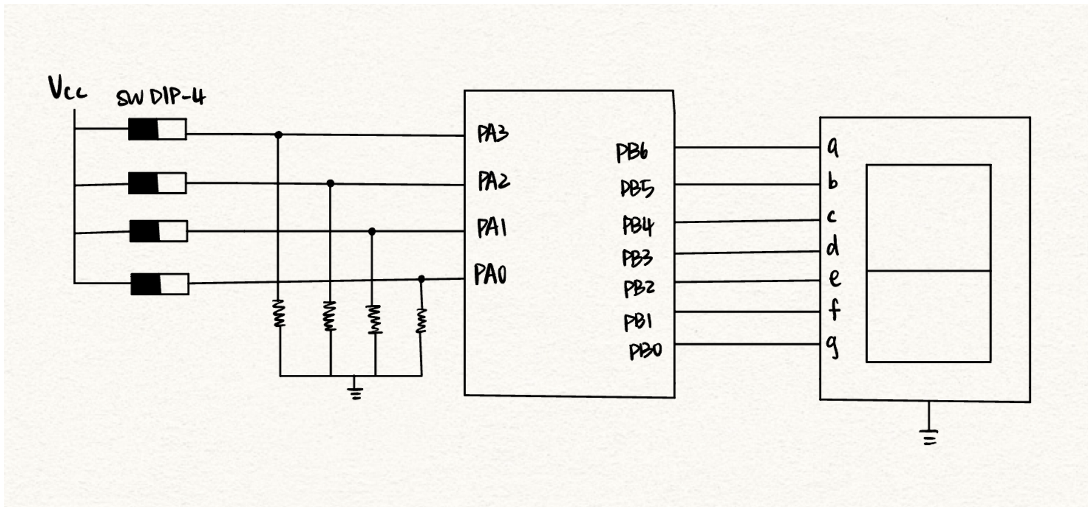


Key	Column	Row	Port A (binary)	Port A (hex)
1	0	0	0b00011000	0x18
2	1	0	0b00101000	0x28
3	2	0	0b01001000	0x48
A	3	0	0b10001000	0x88
4	0	1	0b00010100	0x14
5	1	1	0b00100100	0x24
6	2	1	0b01000100	0x44
B	3	1	0b10000100	0x84
7	0	2	0b00010010	0x12
8	1	2	0b00100010	0x22
9	2	2	0b01000010	0x42
C	3	2	0b10000010	0x82
*	0	3	0b00010001	0x11
0	1	3	0b00100001	0x21
#	2	3	0b01000001	0x41
D	3	3	0b10000001	0x81

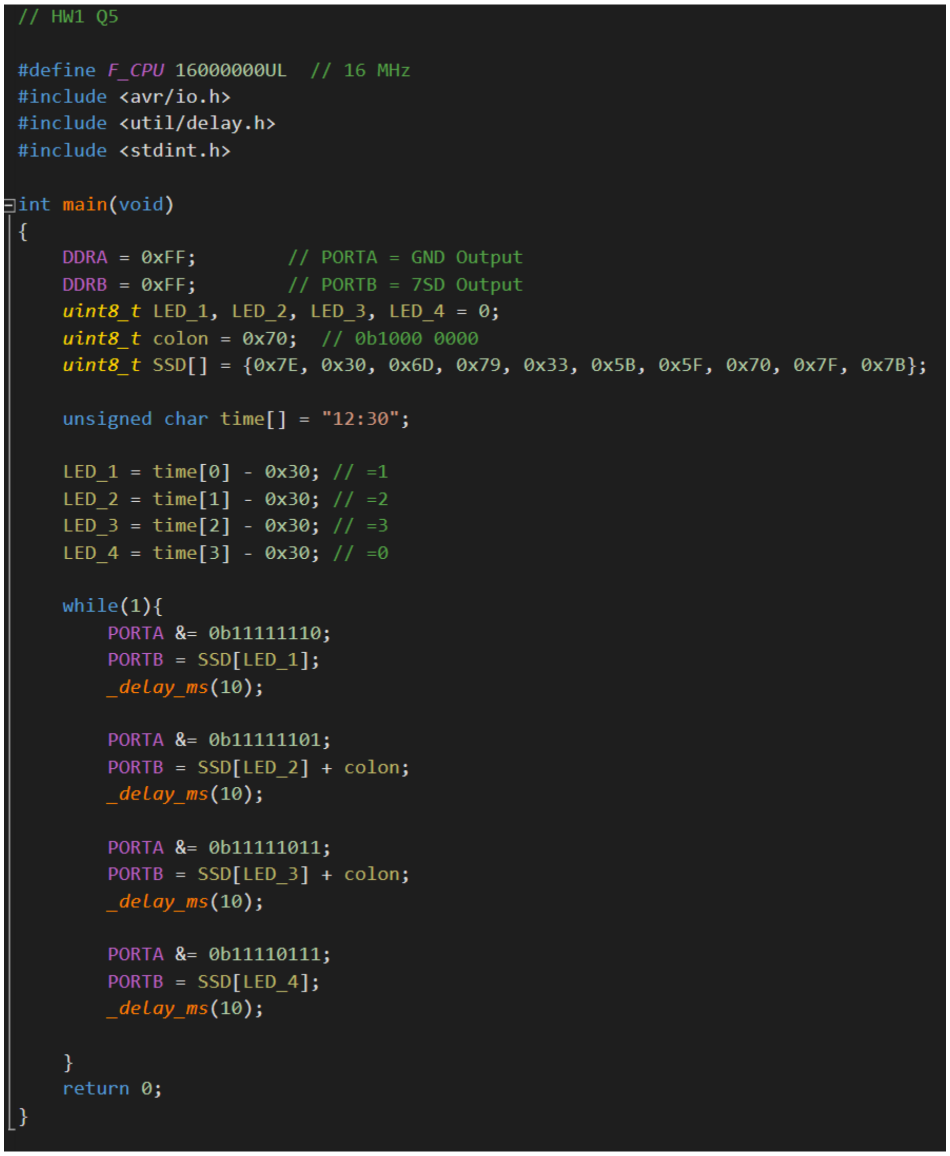
4. A quad DIPS and a common cathode 7SD are interfaced to the microcontroller via PORTA and PORTB respectively. Write a program that will read the DIP switches and convert their settings to hex digit then display that digit on the 7SD. Assume that the 7SD is grounded. Draw the diagram for the circuit and explain where all pins are connected.



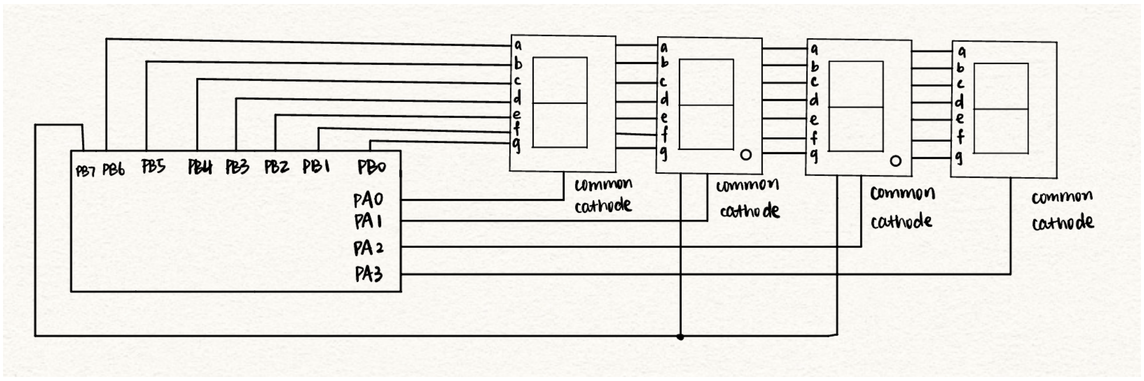
```c
// HW1 Q4

#define F_CPU 16000000UL  // 16 MHz
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>

// Quad DIP = 4 bit --> 0b0000

int main(void)
{
    DDRA = 0;            // PORTA = DIP4 Input
    DDRB = 0xFF;         // PORTB = 7SD Output

    // Unsigned char = uint8 in AVR
    uint8_t SSD[] = {0x7E, 0x30, 0x6D, 0x79, 0x33, 0x5B, 0x5F, 0x70, 0x7F, 0x7B};
    uint8_t DIP4_Value;

    while(1){
        DIP4_Value = PINA;

        if(DIP4_Value < 10){
            PORTB = SSD[DIP4_Value];
        }

        else{
            PORTB = 0;
        }
        _delay_ms(10);
    }
    return 0;
}
```



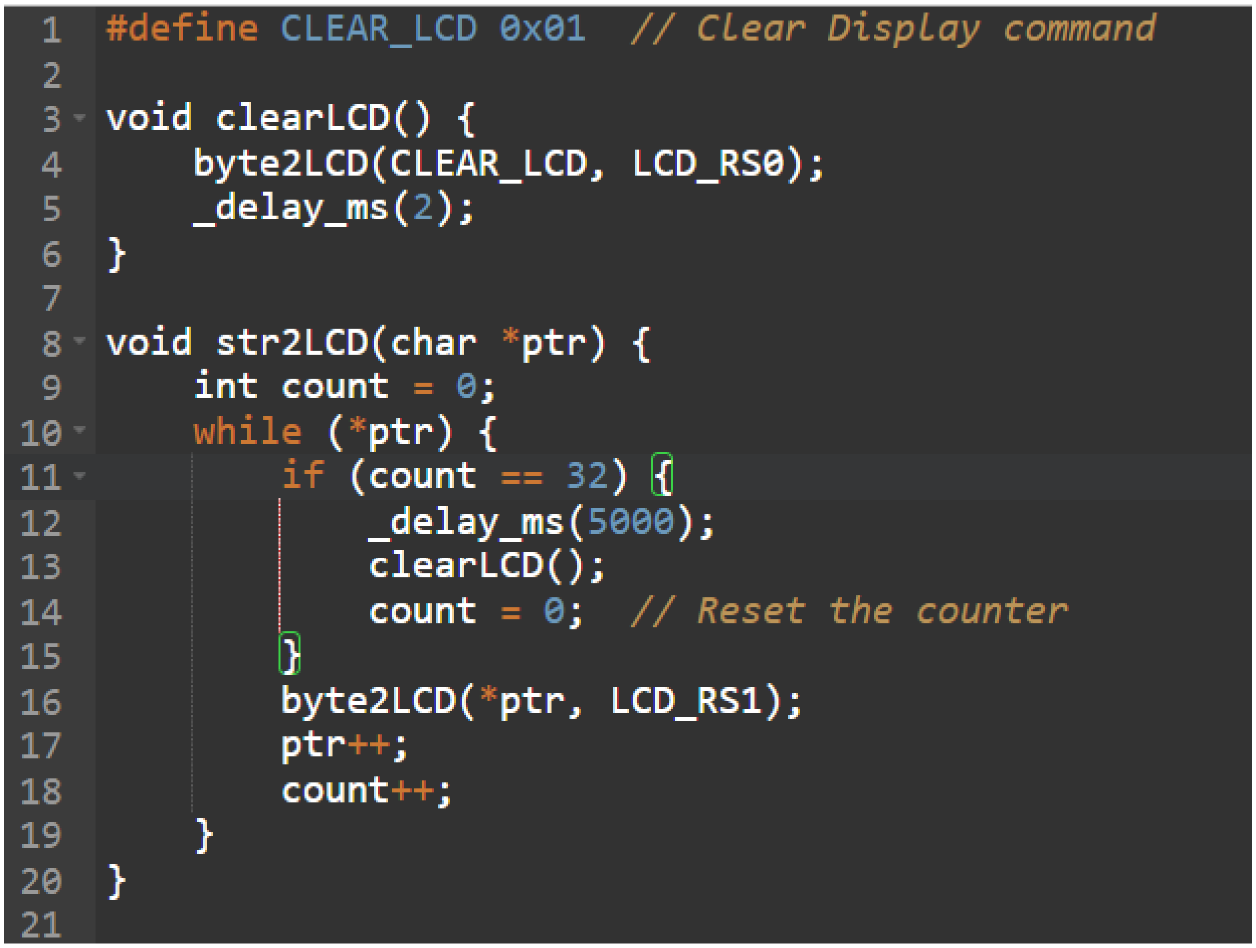
5. The variable time is a string of containing the two digits for the hours and 2 digits for the minutes, e.g. "08:30". The inputs of four common cathode 7SDs are connected to PORTB. The GND pins are connected to PORTA pins3-0. Write a program that will display the time on the 7SDs. Draw a circuit diagram. Be sure to use the dot segment on 2 and 3 7SD.



```c
// HW1 Q5

#define F_CPU 16000000UL  // 16 MHz
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>

int main(void)
{
    DDRA = 0xFF;         // PORTA = GND Output
    DDRB = 0xFF;         // PORTB = 7SD Output
    uint8_t LED_1, LED_2, LED_3, LED_4 = 0;
    uint8_t colon = 0x70;   // 0b1000 0000
    uint8_t SSD[] = {0x7E, 0x30, 0x6D, 0x79, 0x33, 0x5B, 0x5F, 0x70, 0x7F, 0x7B};

    unsigned char time[] = "12:30";

    LED_1 = time[0] - 0x30; // =1
    LED_2 = time[1] - 0x30; // =2
    LED_3 = time[2] - 0x30; // =3
    LED_4 = time[3] - 0x30; // =0

    while(1){
        PORTA &= 0b11111110;
        PORTB = SSD[LED_1];
        _delay_ms(10);

        PORTA &= 0b11111101;
        PORTB = SSD[LED_2] + colon;
        _delay_ms(10);

        PORTA &= 0b11111011;
        PORTB = SSD[LED_3] + colon;
        _delay_ms(10);

        PORTA &= 0b11110111;
        PORTB = SSD[LED_4];
        _delay_ms(10);

    }
    return 0;
}
```



6. Given the 2x16 LCD circuit in the notes, slide 50. Re-write str2lcd function that can be used for messages greater than 32 characters. For messages greater that 32 characters, wait 5 before clearing the screen and continuing to display from the beginning.



```c
#define CLEAR_LCD 0x01  // Clear Display command

void clearLCD() {
    byte2LCD(CLEAR_LCD, LCD_RS0);
    _delay_ms(2);
}

void str2LCD(char *ptr) {
    int count = 0;
    while (*ptr) {
        if (count == 32) {
            _delay_ms(5000);
            clearLCD();
            count = 0;  // Reset the counter
        }
        byte2LCD(*ptr, LCD_RS1);
        ptr++;
        count++;
    }
}
```