1) Write a program which uses INT and Timer 0/1/2/3 interrupts to play the chord C# major for 1.000 seconds when button RB0 is pressed.

C#maj C# F G#

- RA0: C#3 138.59 Hz (Timer 1)
- RA1: F3 174.61 Hz (Timer 2)
- RA2: G#3 207.65 Hz (Timer 3)
- 1.000 second: (Timer 0)
- Rising edge on RB0 (INT0)
Interrupt Service Routine:

```c
void interrupt IntServe(void)
{
    if(INT0) {
        TMR0 = -39062;
        RA3 = 1;
        INTOIF = 0;
    }
    if(TMR0IF) {
        RA3 = 0;
        TMR0IF = 0;
    }
    if(TMR1IF) {
        TMR1 = -36026;
        if(RA3) RA0 = !RA0;
        TMR1IF = 0;
    }
    if(TMR2IF) {
        if(RA3) RA1 = !RA1;
        TMR2IF = 0;
    }
    if(TMR3IF) {
        TMR1 = -24027;
        if(RA3) RA2 = !RA2;
        TMR3IF = 0;
    }
}
```
Interrupt Initialization

// set up Timer0 for PS = 256
T0CS = 0;
T0CON = 0x87;
TMR0ON = 1;
TMR0IE = 1;
TMR0IP = 1;
PEIE = 1;
// set up Timer1 for PS = 1
TMR1CS = 0;
T1CON = 0x81;
TMR1ON = 1;
TMR1IE = 1;
TMR1IP = 1;
PEIE = 1;
// set up Timer2 for 174.61Hz ( N = 28,635 = 7 * 256 * 16 )
T2CON = 0x3F;
PR2 = 255;
TMR2ON = 1;
TMR2IE = 1;
TMR2IP = 1;
PEIE = 1;
// set up Timer3 for PS = 1
TMR3CS = 0;
T3CON = 0x81;
TMR3ON = 1;
TMR3IE = 1;
TMR3IP = 1;
PEIE = 1;
// set up INT0 for rising edges
INT0IE = 1;
TRISB0 = 1;
TRISB1 = 1;
INTEDG0 = 1;
// turn on all interrupts
GIE = 1;

Term Project

2) Specify the requirements for your term project subject to including

- At least two interrupts,
- At least three things we covered in ECE 376 (neopixel, stepper motor, LCD display, keypad, etc.), and
- Some statistical analysis
Insanely Precise Chess Clock:

3) Requirement:

Keep track of two player's times to 0.1ms.

Input: Buttons RB0, RB1, RB2

Output: LCD display

How they relate:
- Press RB1 to start the game. That sets each player's time to 5 minutes (300 seconds)
- When RB0 is pressed (player A), his/her clock stops and player B's clock starts
- When RB1 is pressed (player b), his/her clock stops and player A's clock starts
- When running, a player's clock decrements down to 0:00 and stops at 0.
- The LCD display displays the time to 100us
- Tolerance: 50 clocks

Interrupts Used:
- INT0: Player 1's button (Player 2's clock starts to decrement)
- INT1: Pause the game (both clocks stop)
- INT2: Player 2's button (Player 1's clock starts to decrement)
- Timer2 100us timer (1000 clocks)
- Reset: Return times to 5:00.0000 (not really an interrupt)
4a) **Analysis:** Give computations for setting up Timer interrupts

- Pre-scalars are set to 1 for a resolution of 100ns
- INT interrupts are set up for rising edges

Timer2 = 1000 clocks = 1 * 250 * 4

\[
T2CON = 0x05; \\
PR2 = 249
\]

4b) **C-Code and Flow Chart**

Note:
- It looks like there are five programs running in parallel: the main routine and four interrupts.
- Each interrupt is fairly simple. Interrupt, do something, then get out.

Don't do loops or complex operations inside the interrupt - there may be another interrupt coming shortly.
void interrupt IntServe(void)
{
    if (TMR2IF) {
        if (TURN == 1)
            if (TIME1) TIME1 = TIME1 - 1;
        if (TURN == 2)
            if (TIME2) TIME2 = TIME2 - 1;
        TMR2IF = 0;
    }
    if (INT0IF) {
        TURN = 2;
        INT0IF = 0;
    }
    if (INT1IF) {
        TURN = 0;
        INT1IF = 0;
    }
    if (INT2IF) {
        TURN = 1;
        INT2IF = 0;
    }
}

void LCD_Out(unsigned long int DATA, unsigned char N)
{
    unsigned char A[10], i;
    for (i=0; i<6; i++) {
        A[i] = DATA % 10;
        DATA = DATA / 10;
    }
    for (i=6; i>0; i--) {
        if (i == N) LCD_Write('.');
        LCD_Write(A[i-1] + '0');
    }
}

void main(void)
{
    unsigned int i;
    unsigned long int M1, S1;
    unsigned long int M2, S2;
    TRISA = 0;
    TRISB = 0xFF;
    TRISC = 0x04; // capture every rising edge
    TRISD = 0;
    ADCON1 = 0x0F;
    // Turn on Timer2 for 1000 clocks
    T2CON = 0x05;
    PR2 = 249;
    TMR2ON = 1;
    TMR2IE = 1;
    PEIE = 1;
    // Turn on INT0 for rising edges
    INTEDG0 = 1;
    INT0IE = 1;
    TRISB0 = 1;
    // Turn on INT1 for rising edges
    INTEDG1 = 1;
    INT1IE = 1;
    TRISB1 = 1;
    // Turn on INT2 for rising edges
    INTEDG2 = 1;
    INT2IE = 1;
    TRISB2 = 1;
    LCD_Init();
    Wait_ms(100);
while(1) {
    M1 = TIME1 / 600000;
    S1 = TIME1 % 600000;

    M2 = TIME2 / 600000;
    S2 = TIME2 % 600000;

    LCD_Move(0,4);
    LCD_Write(M2 + 48);
    LCD_Write(':');
    LCD_Out(S2,4);

    LCD_Move(1,4);
    LCD_Write(M1 + 48);
    LCD_Write(':');
    LCD_Out(S1,4);
}

Memory Summary:

<table>
<thead>
<tr>
<th>Space Type</th>
<th>Used</th>
<th>Of Total</th>
<th>As Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program space</td>
<td>A7Eh (2686)</td>
<td>10000h bytes</td>
<td>4.1%</td>
</tr>
<tr>
<td>Data space</td>
<td>6Ah (106)</td>
<td>F80h bytes</td>
<td>2.7%</td>
</tr>
<tr>
<td>EEPROM space</td>
<td>0h (0)</td>
<td>400h bytes</td>
<td>0.0%</td>
</tr>
<tr>
<td>ID Location space</td>
<td>0h (0)</td>
<td>8h nibbles</td>
<td>0.0%</td>
</tr>
<tr>
<td>Configuration bits</td>
<td>0h (0)</td>
<td>7h words</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

The resulting code compiles into 1343 lines of assembler.
It takes up 4.1% of program memory (the PIC can do a lot more)
5) Validation: Include
   - Measurements of the Timer interrupt (is is correct?)
   - Checking that the number generated is in fact random (use a Chi-squared test)

   ![Image of oscilloscope display]

   RA0: Timer2 is interrupting every 100us (check)

   ![Image of circuit board with LCD display]

   Resulting Display

Checking the functionality:
   - Time starts on 5:00.0000 on reset.
   - RB0 starts Player #2's clock. It decrements to zero and stops at zero (shown)
   - RB1 pauses the clock. Both times stop decrementing.
   - RB2 starts Player #1's clock. It decrements to zero and stops at zero.

6) Demo (30pt): In person or with a video.