For the questions 1-2 assume a 1st order LPF with a corner frequency of 5kHz. Answers will be graded based on your derivation of the answer, correct use of units, and the answer. Round answers to 2 decimal places.

1. (5 pts.) What is the output amplitude (in volts) of a 1v, 50kHz sin wave?

   The 50kHz waveform is 1 decade higher than 5kHz
   @ -20dB/decade a 1v signal is attenuated by -20dB
   -20dB = 20log(Vo/1V)   Vo = 0.1

2. (5 pts.) At what frequency would a 1v sin wave become 0.047v at the output?

   The signal attenuation is 20log(0.047/1) = -26.6dB
   @-20dB/decade this amount of attenuation is achieved in -26dB/(-20dB/decade) = 1.33 decades
   In terms of frequency 1.33 decades = log(x/5kHz) x = 107kHz

3. (10 pts.) Given a 0-10kHz signal of interest, a 2nd order LPF and a 16-bit ADC. What sampling rate should you use to eliminate aliases effecting your ADC measurements?

4. (6 pts.) A DDS system updates its output every 10us from a 64-entry ROM containing 1 wavelength using a 6.10 fixed point format phase increment. What is the frequency resolution of the system? State your answer in Hz using dimensional analysis and round your answer to 2 decimal places.
5. (6 pts.) You are computing a function, FUNC, using linear interpolation that takes a 16-bit integer x as input and returns an 16-bit output. The lookup table has 129 entries – the values in the array are immaterial to this problem. Determine the values for SFT1, MASK and SFT2 in the following code.

```c
uint16_t FUNC(uint16_t x) {
    uint16_t lut[129] = {...};
    uint16_t index, delta, frac, base;
    index = x >> SFT1;
    base = lut[index];
    delta = lut[index+1]-lut[index];
    frac = x & MASK;
    return(base+((frac*delta) >> SFT2));
} // end FUNC
```

<table>
<thead>
<tr>
<th></th>
<th>SFT1</th>
<th>MASK</th>
<th>SFT2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>0b0000 0001 1111 1111 = 0x01FF</td>
<td>9</td>
</tr>
</tbody>
</table>

6. (12 pts.) Write C-code for an ISR that collects 8-bit samples from a microphone that is connected to the active ADC channel. The ADC and timer are properly configured. The ISR should:
   - stores the values in a local 8-entry array,
   - increments the array index, rolling it over to 0 every time that the index reaches the end of the array,
   - set a global flag, newSetOfSamples true, every time the array index rolls over to 0,
   - retains the array values between ISR calls,
   - gets the ADC value from the ADRESH register.

In the space below, declare variables needed and write the code to continuously acquire microphone samples into the array.

```c
void tmr0_isr(void) {
    static uint8_t sample[8];
    uint_8 index = 0;
    sample[index] = ADRESH;
    index = index + 1;   // Alternative
    if (index > 7) {
        index = 0;  // Method to increment
        newSetOfSamples = true;
    }
    ADCON0bits.GO_NOT_DONE = 1;  // start a new conversion
    INTCONbits.TMR0IF=0;  // always clear the interrupt flag
    TMR0 = 0xFFFF - 100;  // Next interrupt for 100us
} // end tmr0_isr
```
7. (20 pts.) Determine the values stored at \texttt{varI8} and \texttt{varX8} in the table below as the program executes. Only record variable values when the flow of control reaches the commented lines. You may represent your answers in any base you want. There are more blank rows in the table than you will need.

```assembly
CLRF varX8
MOVLW 5
MOVWF varI8

loop MOVF varI8, W
      ADDWF varX8, F ; RECORD VARIABLE VALUES IN TABLE EVERY TIME THIS LINE IS REACHED
      DECSZ varI8
      GOTO loop

next MOVLW B'10101100'
      ANDWF varX8, F ; RECORD VARIABLE VALUES IN TABLE EVERY TIME THIS LINE IS REACHED

end GOTO end
```

<table>
<thead>
<tr>
<th>\texttt{varI8}</th>
<th>\texttt{varX8}</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5+0=5</td>
</tr>
<tr>
<td>4</td>
<td>5+4=9</td>
</tr>
<tr>
<td>3</td>
<td>9+3=12</td>
</tr>
<tr>
<td>2</td>
<td>12+2=14</td>
</tr>
<tr>
<td>1</td>
<td>14+1=15</td>
</tr>
<tr>
<td>0</td>
<td>0b00001100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembly</th>
<th>Description</th>
<th>Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLRF f, a</td>
<td>Clear F</td>
<td>0110 101a ffff ffff</td>
</tr>
<tr>
<td>MOVLW k</td>
<td>Move literal to WREG</td>
<td>0000 1110 kkkk kkkk</td>
</tr>
<tr>
<td>MOVWF f, a</td>
<td>Move WREG to f</td>
<td>0110 110a ffff ffff</td>
</tr>
<tr>
<td>MOVF f, d, a</td>
<td>Move f</td>
<td>0101 00da ffff ffff</td>
</tr>
<tr>
<td>ADDWF f, d, a</td>
<td>Add WREG and f</td>
<td>0010 01da ffff ffff</td>
</tr>
<tr>
<td>ANDWF f, d, a</td>
<td>AND WREG with f</td>
<td>0001 01da ffff ffff</td>
</tr>
<tr>
<td>DECSZ f, d, a</td>
<td>Decrement f, Skip if 0</td>
<td>0010 11da ffff ffff</td>
</tr>
</tbody>
</table>