Example of how to comment SE423 Lab/Homework Code. For Lab, lab partners could use both of their initials
You should submit your C file with the comments. Do not create a Word document like this example does.
This format for commenting does not have to be done for HW1 Question 1 (C review).
I made the initials Bold for emphasis. You do not need to bold your initials.

```
// Andy Jones is commenting this Code.  Comments marked by my initials AAJ
//##########################################################################################
// FILE:   LABstarter_main.c
//
// TITLE:  Lab Starter
//##########################################################################################
#include <stdio.h>
#include <stdlib.h>
....  Omitted Code
....  You should not omit code in your submission, I am here so this commenting example fits on one page
....
uint32_t numTimer0calls = 0;
uint32_t numSWIcalls = 0;
extern uint32_t numRXA;
uint16_t UARTPrint = 0;
uint16_t LEDdisplaynum = 0;
float Kp = 3.4;  // AAJ Kp and Ki gains used for the PI controller.  Hand tuned for desired response.
float Ki = 29.3; // I did find that a number of gain combinations worked, but settled on these gains.
float Vref = 2.5;
float Vactual = 0;
float Ik = 0;
float Vout = 0;

void main(void)
{
   // PLL, WatchDog, enable Peripheral Clocks
   // This example function is found in the F2837xD_SysCtrl.c file.
   InitSysCtrl();
....  Omitted Code
....  You should not omit code in your submission, I am here so this commenting example fits on one page
....
   // Configure CPU-Timer 0, 1, and 2 to interrupt every second:
   // 200MHz CPU Freq, 1 second Period (in uSeconds)
   ConfigCpuTimer(&CpuTimer0, 200, 5000);  //AAJ  Set CPU Timer 0 to call its interrupt function every 5ms.  This is the
                                           // specified sample rate for the control loop calculations.
   ConfigCpuTimer(&CpuTimer1, 200, 20000);
   ConfigCpuTimer(&CpuTimer2, 200, 40000);
....  Omitted Code
....  You should not omit code in your submission, I am here so this commenting example fits on one page
....
   // Enable global Interrupts and higher priority real-time debug events
   EINT;  // Enable Global interrupt INTM
   ERTM;  // Enable Global realtime interrupt DBGM
```

!!!! There is a second page !!!!!

```
   // IDLE loop. Just sit and loop forever
   while(1)
   {
        if (UARTPrint == 1 ) {
// AAJ Here I am using GPIO67 to see how much time the serial_printf function takes to run.  In lab I will connect the Oscilloscope
// to GPIO67's pin and measure the width of the pulse created by setting GPIO67 high (3.3Volts) before the function is called and then
// setting GPIO67 low (0Volts) after the function is completed.  GPIO67 is controlled by the GPC registers.  GPIO67 is the 4th bit
// in the GPC registers.  Bit 0 is for GPIO64 Bit 1 is for GPIO65, Bit 2 is for GPIO66, Bit 3 is for GPIO67, Bit 4 is for GPIO68 and so on.
// The below bitfield C statement hides the bit number from me and allows me to just set the associated bit pertaining to GPIO67.
// Setting a bit to 1 in the GPCSET register causes that associated output pin to be high (3.3 Volts).
// Setting a bit to 0 in the GPCSET register does nothing.
        GpioDataRegs.GPCSET.bit.GPIO67 = 1;
        //AAJ  Printing Desired and Actual Speed  %.3f tells printf to print 3 decimal places of float variable
        serial_printf(&SerialA,"%.3f Desired Speed, %.3f Actual Speed\r\n",Vref,Vactual);
        UARTPrint = 0;
        // AAJ  Set GPIO67 to low (0Volts) so Oscilloscope and see when serial_printf function is complete
        // Setting a bit to 1 in the GPCCLEAR register causes that associated output pin to be low (0 Volts).  Setting a bit
        // to 0 in the GPCCLEAR register does nothing.
        GpioDataRegs.GPCCLEAR.bit.GPIO67 = 1
    }
   }
}

// cpu_timer0_isr - CPU Timer0 ISR
__interrupt void cpu_timer0_isr(void)
{
        Vactual = readfeedback();  // AAJ this is a dummy function for this example.

        // AAJ Here I am calculating the PI control law.  I am using the "forward rule" to approximate the integral
        // Then I calculate the control effort Vout which is Kp times the error between desired speed and actual speed
        // and Ki times the integral appoximation of the error.  I also check for integral windup and then saturate
        // the output between -10 and +10.
        Ik = Ikold + (Vref-Vactual)*0.005;
        Vout = Kp*(Vref-Vactual) + Ki*Ik;

        if (fabs(Vout) > 10) Ik = Ikold;
        if (Vout > 10) Vout = 10;
        if (Vout < -10) Vout = -10;

        outPWM(Vout);   //AAJ dummy function to output control

        Ikold = Ik;  //AAJ save past state for next 5ms

        CpuTimer0.InterruptCount++;

        numTimer0calls++;

        if ((numTimer0calls%20) == 0) {
                UARTPrint = 1;  //AAJ Print to Tera Term every 100ms  5ms * 20 = 100ms

                // Blink LaunchPad Red LED
                GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;
        }

        // Acknowledge this interrupt to receive more interrupts from group 1
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}
....  Omitted Code
....  You should not omit code in your submission, I am so this commenting example fits on one page
....
```