Task 00: Execute supplied code

```
//
/*
 * Melissa Cordova
 * CPE 403 - LAB 5
 *
 * main.c
 */

#include <stdint.h>     //variable deifinitions for the C99 standard
#include <stdbool.h>    //boolean definitions for the C99 standard
#include "inc/hw_memmap.h"  //macros defining the memory map of the TivaC
#include "inc/hw_types.h"   //define common types and macros
#include "driverlib/debug.h"    //defines debug
#include "driverlib/sysctl.h"   //defines and macros for System Control API
#include "driverlib/adc.h"  //definitions for using the ADC driver
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"

#ifdef DEBUG
void__error__(char *pcFilename, uint32_t u132lLine)
{

}
#endif

int main(void)
{
    uint32_t ui32ADC0Value[4];  //create an array to store data from ADC FIFO
                                //with depth of 4

    volatile uint32_t ui32TempAvg;  //sorts the average of the temp
    volatile uint32_t ui32TempValueC;   //store the Celsius temp
    volatile uint32_t ui32TempValueF;   //store the Fahrenheit temp

    //system clock to run at 40MHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //enable the ADC0 peripheral
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    //ADC sequencer
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

    //Configuring steps 0-2 on sequencer 1 to sample temp sensor
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
```

```
    //Sample temp sensor and configure interrupt flag to set
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    //enable ADC
    ROM_ADCSequenceEnable(ADC0_BASE, 1);

    while(1)
    {
        //ADC conversion
        ROM_ADCIntClear(ADC0_BASE, 1);
        //ADC conversion with software
        ROM_ADCProcessorTrigger(ADC0_BASE, 1);

        while(!ADCIntStatus(ADC0_BASE, 1, false))
        {

        }

        //ADC value from ADC Sampler Sequence
        ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);

        //Average of temp sensor data
        ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;

        //calculate celsius value of temp
        ui32TempValueC = (1475- ((2475*ui32TempAvg)) / 4096) /10;

        //calculate Fahrenheit temp
        ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;


    }

}
```

Task 01: Change the ADC Sequencer to SS2. Turn on the LED at PF3 if the temperature is greater than 75 degF. Use internal temperature sensor for all SS2 sequence.

```
#include <stdint.h>     //variable deifinitions for the C99 standard
#include <stdbool.h>    //boolean definitions for the C99 standard
#include "inc/hw_memmap.h"  //macros defining the memory map of the TivaC
#include "inc/hw_types.h"   //define common types and macros
#include "driverlib/sysctl.h"   //defines and macros for System Control API
#include "driverlib/adc.h"  //definitions for using the ADC driver
#include "driverlib/gpio.h" //defines macros for GPIO
#include "driverlib/interrupt.h"    //defines and macros for NVIC
#include "driverlib/timer.h" //defines and macros for Timer API
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"
```

```
.
.
.
ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //enable the ADC0 peripheral
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    //ADC sequencer
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);

    //Configuring steps 0-2 on sequencer 2 to sample temp sensor
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);

    //Sample temp sensor and configure interrupt flag to set
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    //enable ADC
    ROM_ADCSequenceEnable(ADC0_BASE, 2);

    while(1)
    {
        //ADC conversion
        ROM_ADCIntClear(ADC0_BASE, 2);
        //ADC conversion with software
        ROM_ADCProcessorTrigger(ADC0_BASE, 2);

        while(!ADCIntStatus(ADC0_BASE, 2, false))
        {

        }

        //ADC value from ADC Sampler Sequence
        ROM_ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);

              .
              .
              .
        if(ui32TempValueF > 75)  //if temp is > 75 degrees F
        {
            //Turn on LED at PF1 because could not get it to turn on PF3
            ROM_GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);
        }
        else
        {
            //Turn off LED
            ROM_GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
        }
```

```
    }

}
```

Task 02: Introduce hardware averaging to 32. Using the timer TIMER1A conduct an ADC conversion on overflow every 0.5sex. Use the TIMER1A interrupt.

```
#include <stdint.h>     //variable deifinitions for the C99 standard
#include <stdbool.h>    //boolean definitions for the C99 standard
#include "inc/hw_memmap.h"  //macros defining the memory map of the TivaC
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_types.h"   //define common types and macros
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"   //defines and macros for System Control API
#include "driverlib/adc.h"  //definitions for using the ADC driver
#include "driverlib/gpio.h" //defines macros for GPIO
#include "driverlib/interrupt.h"    //defines and macros for NVIC
#include "driverlib/timer.h" //defines and macros for Timer API
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"


int main(void)
{
    uint32_t ui32Period;


        .
        .
        .
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);

    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

     //calculate and set delay
    ui32Period = SysCtlClockGet()/2;
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period-1);

    ADCHardwareOversampleConfigure(ADC0_BASE, 32);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1| GPIO_PIN_2| GPIO_PIN_3);

        .
        .
        .
    //enable interrupts
    IntEnable(INT_TIMER1A);
    //set timer 1 to interrupt
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();
    //enable timer
    TimerEnable(TIMER1_BASE, TIMER_A);
```

```
    while(1)
    {

    }
}
void IntTimer1Handler(void)
{


        TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
         //ADC conversion
        .

        .

        .
```