Ameera Essaqi
Github root directory: https://github.com/AmeeraE/microcontrollers/tree/master/TIVAC

Date Submitted: 12/11/19

Task 00: Execute provided code

Youtube Link: **https://youtu.be/-j3UURyLUjY**

Task 01:

Youtube Link: **https://youtu.be/McupezwJ0lk**

Modified Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "driverlib/adc.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/timer.h"
#include "driverlib/debug.h"

void PrintUART(void);
void configTimer1A(void);
void convertUARTtemp(uint32_t);
void UART_OutChar(char);


uint32_t halfPeriod;
uint32_t ui32ADC0Value[1];
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

int main(void)
{
    // set up system clock
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
    // Enable the URAT0 and GPIOA peripherals
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    // configure the pins for the receiver and transmitter using GPIOPinConfigure
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable GPIO port for LED
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3); //enable pin for LED
    //  Initialize the parameters for the UART: 115200, 8-1-N
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
                        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);// enable the ADC0 peripheral

    ADCHardwareOversampleConfigure(ADC0_BASE, 32); // hardware averaging

    //configure the ADC sequencer.
    ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE,3,0,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    halfPeriod = SysCtlClockGet() / 2 ;

    configTimer1A();
    ADCIntEnable(ADC0_BASE,3);
    ADCSequenceEnable(ADC0_BASE,3);

    while (1)
    {
    }
}
void Timer1IntHandler(void)// add to startup_ccs
{
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    TimerLoadSet(TIMER1_BASE, TIMER_A, halfPeriod);
    ADCIntClear(ADC0_BASE, 3);
    //ADC conversion
    ADCProcessorTrigger(ADC0_BASE, 3);
    // Wait for conversion
    while(!ADCIntStatus(ADC0_BASE, 3, false))
    {
    }

    ADCSequenceDataGet(ADC0_BASE, 3, ui32ADC0Value);
    //  Calculate the average of temperature
    ui32TempAvg = ui32ADC0Value[0];

    // Calculate the Celsius value of the temperature
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;

    //C to F:  F = ( C * 9)/5 +32.
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

    //Print on Terminal
    PrintUART();
}


void configTimer1A(void)
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
    TimerLoadSet(TIMER1_BASE, TIMER_A, halfPeriod);
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

    TimerEnable(TIMER1_BASE, TIMER_A);
    IntMasterEnable(); //enable processor interrupts
}

void PrintUART()
{
    //UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT); //only enable RX and TX interrupts
    UARTCharPut(UART0_BASE, 'T');
    UARTCharPut(UART0_BASE, 'e');
    UARTCharPut(UART0_BASE, 'm');
    UARTCharPut(UART0_BASE, 'p');
    UARTCharPut(UART0_BASE, ':');
    UARTCharPut(UART0_BASE, ' ');
    convertUARTtemp(ui32TempValueF);
    UARTCharPut(UART0_BASE, 'F');
    // linefeed
    UARTCharPut(UART0_BASE, '\n');
    // carriage-return
    UARTCharPut(UART0_BASE, '\r');

}
void convertUARTtemp(uint32_t tempF)
{
    if (tempF >= 10)
    {
        convertUARTtemp(tempF/10);
        tempF %= 10;
    }
    UART_OutChar(tempF + '0');
}

void UART_OutChar(char val)
{
    while((UART0_FR_R & UART_FR_TXFF) != 0);
    UART0_DR_R = val;
}
```

Task 02:

Youtube Link: **https://youtu.be/mHxLuwLw9HM**

Modified Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "driverlib/adc.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/timer.h"
#include "driverlib/debug.h"

#define RED     2 // Value to turn red
#define BLUE    4 // Value to turn blue
#define GREEN   8 // Value to turn green

void PrintCommand(void);
void RedON(void);
void BlueON(void);
void GreenON(void);

void RedOFF(void);
void BlueOFF(void);
void GreenOFF(void);

void PrintTemp(void);
void convertUARTtemp(uint32_t);
void UART_OutChar(char);


uint32_t ui32ADC0Value[1];
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;
char letter;


int main(void)
{
    // set up system clock
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
    // Enable the URAT0 and GPIOA peripherals
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    // configure the pins for the receiver and transmitter using GPIOPinConfigure
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable GPIO port for LED
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3); //enable pin for LED
    //enable pin for LED
    //  Initialize the parameters for the UART: 115200, 8-1-N
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
                        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);// enable the ADC0 peripheral

    ADCHardwareOversampleConfigure(ADC0_BASE, 32); // hardware averaging

     //configure the ADC sequencer
    ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE,3,0,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    IntMasterEnable(); //enable processor interrupts
    IntEnable(INT_UART0); //enable the UART interrupt
    UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT); //only enable RX and TX interrupts
    ADCIntEnable(ADC0_BASE,3);
    ADCSequenceEnable(ADC0_BASE,3);
    PrintCommand();

    while (1)
    {
    }
}

void UARTIntHandler(void)
{
    uint32_t ui32Status;
    ui32Status = UARTIntStatus(UART0_BASE, true);
    UARTIntClear(UART0_BASE, ui32Status);

    letter = UARTCharGet(UART0_BASE);
    UARTCharPut(UART0_BASE, letter);
    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');

    //Checks each case letter case when pressed
    switch(letter)
    {
    case 'R' :
        RedON();
        break;
    case 'B' :
        BlueON();
        break;
    case 'G' :
        GreenON();
        break;
    case 'r' :
        RedOFF();
        break;
    case 'b' :
        BlueOFF();
        break;
    case 'g' :
        GreenOFF();
        break;
    case 'T' :
        PrintTemp();
        break;
    default :
        PrintCommand();
    }
```

```
}

void PrintCommand(void)
{

    UARTCharPut(UART0_BASE, 'E');
    UARTCharPut(UART0_BASE, 'n');
    UARTCharPut(UART0_BASE, 't');
    UARTCharPut(UART0_BASE, 'e');
    UARTCharPut(UART0_BASE, 'r');
    UARTCharPut(UART0_BASE, ' ');
    UARTCharPut(UART0_BASE, 'K');
    UARTCharPut(UART0_BASE, 'e');
    UARTCharPut(UART0_BASE, 'y');
    UARTCharPut(UART0_BASE, ' ');
    UARTCharPut(UART0_BASE, 'C');
    UARTCharPut(UART0_BASE, 'm');
    UARTCharPut(UART0_BASE, 'd');
    UARTCharPut(UART0_BASE, ':');
    UARTCharPut(UART0_BASE, ' ');
}
void RedON(void)
{
    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, RED); //blink red LED
    SysCtlDelay(100000);
    PrintCommand();
}
void BlueON(void)
{
    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, BLUE); //blink LED
    SysCtlDelay(100000);
    PrintCommand();
}
void GreenON(void)
{
    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, GREEN); //blink LED
    SysCtlDelay(100000);
    PrintCommand();
}

void RedOFF(void)
{
    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0); //turn off LED
    SysCtlDelay(100000);
    PrintCommand();
}
void BlueOFF(void)
{
    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0); //turn off LED
    SysCtlDelay(100000);
    PrintCommand();
}

void GreenOFF(void)
```

```
{
    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0); //turn off LED
    SysCtlDelay(100000);
    PrintCommand();
}
void PrintTemp(void)
{
    ADCIntClear(ADC0_BASE, 3);
    // Trigger the ADC conversion with software
    ADCProcessorTrigger(ADC0_BASE, 3);
    // Wait for the conversion to complete.
    while(!ADCIntStatus(ADC0_BASE, 3, false))
    {
    }

    ADCSequenceDataGet(ADC0_BASE, 3, ui32ADC0Value);
    //average of the temperature data

    ui32TempAvg = ui32ADC0Value[0];

    // Calculate Celsius temperature
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;

    // Celsius to Fahrenheit: F = ( C * 9)/5 +32
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
    UARTCharPut(UART0_BASE, 'T');
    UARTCharPut(UART0_BASE, 'e');
    UARTCharPut(UART0_BASE, 'm');
    UARTCharPut(UART0_BASE, 'p');
    UARTCharPut(UART0_BASE, ':');
    UARTCharPut(UART0_BASE, ' ');
    convertUARTtemp(ui32TempValueF);
    UARTCharPut(UART0_BASE, 'F');

    UARTCharPut(UART0_BASE, '\n');
    UARTCharPut(UART0_BASE, '\r');
    PrintCommand();
}

void convertUARTtemp(uint32_t tempF)
{
    if (tempF >= 10)
    {
        convertUARTtemp(tempF/10);
        tempF %= 10;
    }
    UART_OutChar(tempF + '0');
}

void UART_OutChar(char val)
{
    while((UART0_FR_R & UART_FR_TXFF) != 0);
    UART0_DR_R = val;
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.