Task 00: Execute supplied code

```
//
/* Melissa Cordova
 * CPE 403 - LAB 4
 *
 * main.c
 */

#include <stdint.h>         //variable definitions for the C99 standard
#include <stdbool.h>        //boolean definitions for the C99 standard
#include "inc/tm4c123gh6pm.h"   //definitions for the interrupt and register assig
#include "inc/hw_memmap.h"      //macros defining the memory map of the TivaC
#include "inc/hw_types.h"       //defines common types and macros
#include "driverlib/sysctl.h"   //defines and macros for System Control API
#include "driverlib/interrupt.h"    //defines and macros for NVIC Controller
#include "driverlib/gpio.h"     //Defines and macros for GPIO API of DriverLib
#include "driverlib/timer.h"    //defines and macros for Timer API of DriverLib

int main (void)
{
    uint32_t ui32Period;        //variable ui32Period with unsigned 32-bit int
    //system clock to run at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);        //enable GPIO peripheral
    //configure pins
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    //configures Timer 0 as a 32-bit timer in periodic mode
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    //toggle GPIO at 10Hz and a 50% duty cycle and interrupt at 1/2 period
    ui32Period = (SysCtlClockGet()/10) /2;
    //load into Timer's Interval Load register
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period-1);

    IntEnable(INT_TIMER0A);  //enable specific vector associated with Timer0A
    //master interrupt enable API for all interrupts
    TimerIntEnable(TIMER0_BASE,TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    TimerEnable(TIMER0_BASE, TIMER_A);  //enable the timer

    while(1)
    {

    }

}

void Timer0IntHandler(void)
{
    //Clear the timer interrupt
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    //Read the current state of the GPIO pin and
    //write back the opposites state

    if(GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_2))
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
     }
     else
     {
         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
     }

}
```

Task 01: Change the toggle of the GPIO at 50Hz and at 75% duty cycle and verify

```
#include <stdint.h>         //variable definitions for the C99 standard
#include <stdbool.h>        //boolean definitions for the C99 standard
#include "inc/tm4c123gh6pm.h"   //definitions for the interrupt and register assig
#include "inc/hw_memmap.h"      //macros defining the memory map of the TivaC
#include "inc/hw_types.h"       //defines common types and macros
#include "driverlib/sysctl.h"   //defines and macros for System Control API
#include "driverlib/interrupt.h"    //defines and macros for NVIC Controller
#include "driverlib/gpio.h"     //Defines and macros for GPIO API of DriverLib
#include "driverlib/timer.h"    //defines and macros for Timer API of DriverLib

int main (void)
{

    uint32_t ui32Period;        //variable ui32Period with unsigned 32-bit int

    //system clock to run at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);        //enable GPIO peripheral
    //configure pins
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    //configures Timer 0 as a 32-bit timer in periodic mode
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    //toggle GPIO at 50Hz and at 75% duty cycle
    ui32Period = (SysCtlClockGet()/50) /2;
    //load into Timer's Interval Load register
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period-1);

    IntEnable(INT_TIMER0A);  //enable specific vector associated with Timer0A
    //master interrupt enable API for all interrupts
    TimerIntEnable(TIMER0_BASE,TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    TimerEnable(TIMER0_BASE, TIMER_A);  //enable the timer

    while(1)
    {
    }

}

void Timer0IntHandler(void)
{
    //Clear the timer interrupt
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

    //Read the current state of the GPIO pin and
    //write back the opposites state

    if(GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_2))
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
     }
     else
     {
         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
         //75% duty cycle
         SysCtlDelay(2000000);
     }

}
```

Task 02: Include a GPIO Interrupt to task 02 from switch SW2 to turn ON and the LED for 2 sec. Use a Timer1 to calculate the 2 sec delay. The toggle of the GPIO is suspended when executing the interrupt.

```
#include <stdint.h>         //variable definitions for the C99 standard
#include <stdbool.h>        //boolean definitions for the C99 standard
#include "inc/tm4c123gh6pm.h"   //definitions for the interrupt and register assig
#include "inc/hw_memmap.h"      //macros defining the memory map of the TivaC
#include "inc/hw_types.h"       //defines common types and macros
#include "driverlib/sysctl.h"   //defines and macros for System Control API
#include "driverlib/interrupt.h"    //defines and macros for NVIC Controller
#include "driverlib/gpio.h"     //Defines and macros for GPIO API of DriverLib
#include "driverlib/timer.h"    //defines and macros for Timer API of DriverLib

void IntGPIOF0(void);

int main (void)
{

    uint32_t ui32Period;        //variable ui32Period with unsigned 32-bit int

    //system clock to run at 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);        //enable GPIO peripheral
    //configure pins
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    //configures Timer 0 as a 32-bit timer in periodic mode
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    //toggle GPIO at 50Hz and at 75% duty cycle
    ui32Period = (SysCtlClockGet()/50) /2;
    //load into Timer's Interval Load register
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period-1);

    //Unlock PINF0 to use interrupt for SW2
    SYSCTL_RCGC2_R |= 0x00000020;    //activate clock
    GPIO_PORTF_LOCK_R = 0x4C4F434B;   // GPIO Port F
    GPIO_PORTF_CR_R = 0x1F;
    GPIO_PORTF_AMSEL_R = 0x00;        // disable analog
    GPIO_PORTF_PCTL_R = 0x00000000;
    GPIO_PORTF_DIR_R = 0x0E;          // set PF0 in, PF4, and PF3-1 out
    GPIO_PORTF_AFSEL_R = 0x00;        // disable PF7-0
    GPIO_PORTF_PUR_R = 0x11;          // enable PF0 and PF4
    GPIO_PORTF_DEN_R = 0x1F;          // enable digital I/O

    //register the interrupt handler for PF0
    GPIOIntRegister(GPIO_PORTF_BASE, IntGPIOF0);
    //SW2 goes low when pressed
    GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_FALLING_EDGE);
    //enable interrupts on PF0
    GPIOIntEnable(GPIO_PORTF_BASE, GPIO_PIN_0);


    IntEnable(INT_TIMER0A);  //enable specific vector associated with Timer0A
    //master interrupt enable API for all interrupts
    TimerIntEnable(TIMER0_BASE,TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    TimerEnable(TIMER0_BASE, TIMER_A);  //enable the timer

    while(1)
    {

    }

}

void Timer0IntHandler(void)
{
    //Clear the timer interrupt
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

    //Read the current state of the GPIO pin and
    //write back the opposites state

    if(GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_2))
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

     }
     else
     {
         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
         SysCtlDelay(2000000);
     }

}

void IntGPIOF0(void)
{
    uint32_t delay;

        //clear interrupt flag on pin F0
        GPIOIntClear(GPIO_PORTF_BASE, GPIO_PIN_0);
        //Turn on Blue LED
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4);
        //Enable TIMER1 peripheral
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
        //Set TIMER1 to periodic mode
        TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
        delay = (SysCtlClockGet()/2);
        TimerLoadSet(TIMER1_BASE, TIMER_A, (delay-1));
        TimerEnable(TIMER1_BASE, TIMER_A);
        while (TimerValueGet(TIMER1_BASE, TIMER_A) < (delay-2));
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0 );


}
```