Tenniel Takenaka-Fuller
Github root directory: https://github.com/TennielTakenaka/CLASS2/tree/master/lab6

Date Submitted: **10/12/19**

Task 00: Execute provided code

Youtube Link: https://youtu.be/-9aS85GRhTQ

Task 01:

Youtube Link: https://youtu.be/CU1MILIgQOg

Modified Code:

```
#include<stdint.h>
#include<stdbool.h>
#include"inc/hw_memmap.h"
#include"inc/hw_types.h"
#include"driverlib/sysctl.h"
#include"driverlib/gpio.h"
#include"driverlib/debug.h"
#include"driverlib/pwm.h"
#include"driverlib/pin_map.h"
#include"inc/hw_gpio.h"
#include"driverlib/rom.h"

#define PWM_FREQUENCY 50 //sets a period of 20ms

int main (void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 75; //start position


ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);

    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);
```

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

```
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);

    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 2000); //modify
duty cycle
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);

    while(1)
    {
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
            ui8Adjust--;
            if (ui8Adjust < 50) //0 degrees start
            {
                ui8Adjust = 50;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 2000);
//modify duty cycle for 180 degrees
        }
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            ui8Adjust++;
            if (ui8Adjust > 100) //180 degrees movement
            {
                ui8Adjust = 100;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 500);
//modify duty cycle for 180 degrees
        }
        ROM_SysCtlDelay(100000);
    }

}
```

Task 02:

Youtube Link: https://youtu.be/4aqMX-3O7cc

Modified Code:

```
#include<stdint.h>
#include<stdbool.h>
#include"inc/hw_memmap.h"
#include"inc/hw_types.h"
#include"driverlib/sysctl.h"
#include"driverlib/gpio.h"
#include"driverlib/debug.h"
#include"driverlib/pwm.h"
#include"driverlib/pin_map.h"
#include"inc/hw_gpio.h"
#include"driverlib/rom.h"
```

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

```
#define PWM_FREQUENCY 100 //new frequency

int main (void)
{
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 1;


ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1); //enable pwm
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable switches and red led

    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1); //turns red LED on
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);

    //code to allow use of switches
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN); //configure write to
LED
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);

    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 100); //modify
duty cycle
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2);

    while(1)
    {
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
            ui8Adjust--;
            if (ui8Adjust < 10) //10% duty cycle
            {
                ui8Adjust = 10;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 100);
        }
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            ui8Adjust++;
            if (ui8Adjust > 90) //90% duty cycle
            {
```

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

```
                ui8Adjust = 90;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 100);
        }
        ROM_SysCtlDelay(100000);
    }

}
```

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