Written by: Darrel Januardi

Challenge 1: PWM

The following code was run on MBED simulator:

```
#include "mbed.h"
PwmOut myled(p21);
float duty=0;

int main() {

  myled.period(1.0/10000);    //10 kHz

  while (1) {

   duty=duty+0.1;  // 0 - 1 value. 10% jumps
   if (duty>=1) (duty = 0); // if you get over 100%, zero it.
   myled.write(duty);
   wait(1); //wait a second before adding 10% to the duty cycle

  }

}
```

A PWM output called “myled” is connected to pin 21. It’s period is defined as $\frac{1}{10{,}000}$ seconds, which results in a frequency of 10kHz. While (1) creates an infinite loop, where the variable “duty” increments by 0.1 up to 1 then resets back to 0.

“myled.write(duty)” outputs the duty cycle at every iteration of the loop to “myled”

The following code is the default code provided by MBED simulator for PwmOut:

```
#include "mbed.h"

PwmOut led(p5);

int main() {

    while(1) {

        led = led + 0.10;

        printf("LED is now %.2f\n", led.read());

        wait(0.2);


        if(led == 1.0) {

            led = 0;

        }

    }

    }
```

The output led is connected to pin 5. In the infinite loop, the “led” variable increments by 0.10 at every iteration, up to 1 then resets to 0. The output duty cycle is then printed at every iteration. This code runs similarly to the code above.

The following code is the default code provided by MBED simulator for Pwm Speaker:

```
#include "mbed.h"


PwmOut speaker(p21);


void play_tone(float frequency, float volume, int interval, int rest) {
    speaker.period(1.0 / frequency);
    speaker = volume;
    wait(interval);
    speaker = 0.0;
    wait(rest);
}


int main()
{
    while(1) {
        play_tone(200.0, 0.5, 1, 0);
        play_tone(150.0, 0.5, 1, 0);
        play_tone(125.0, 0.5, 1, 2);
    }


}
```

PWM is used to control the frequency of an audio signal given out by a speaker(pin 21).