Class 5 PWM Task Program

```
/*
 * File:   main.c
 * Author: sagar
 *
 * Created on 1 April, 2025, 3:34 PM
 */

// CONFIG
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)


#include <xc.h>
#include <stdint.h>

#define _XTAL_FREQ 2000000  // Define oscillator frequency for delay calculations

// Function prototypes
void pwmInit(void);
void pwmDutyCycleUpdate(void);

// Global variables for different duty cycles
uint8_t pwm20_lower, pwm20_upper;
uint8_t pwm60_lower, pwm60_upper;
uint8_t pwm90_lower, pwm90_upper;

void main(void)
{
    pwmInit();  // Initialize PWM

    while (1)
    {
        pwmDutyCycleUpdate();  // Continuously update duty cycle
    }
}
```

```
void pwmInit(void)
{
   TRISC = 0xFB;  // Set RC2 (CCP1) as an output pin
   CCP1CON = 0x0C;  // Configure CCP1 module for PWM mode (10-bit resolution)
   T2CON = 0x06;  // Enable Timer2 with a prescaler of 1:16
   PR2 = 0x2F;  // Set PWM period

   // Duty cycle values for 20%, 60%, and 90%
   pwm20_lower = 0x02;
   pwm20_upper = 0x09;

   pwm60_lower = 0x01;
   pwm60_upper = 0x1C;

   pwm90_lower = 0x01;
   pwm90_upper = 0x2A;

   TMR2 = 0;  // Reset Timer2 counter
   TMR2ON = 1;  // Start Timer2
}

void pwmDutyCycleUpdate(void)
{
   /********** 20% duty cycle **********/
   CCPR1L = pwm20_upper;  // Load upper 8 bits of duty cycle
   CCP1CON &= ~(0x3 << 4);  // Clear lower 2 bits
   CCP1CON |= (pwm20_lower << 4);  // Set lower 2 bits
   __delay_ms(3000);  // Wait for 3 seconds

   /********** 60% duty cycle **********/
   CCPR1L = pwm60_upper;
   CCP1CON &= ~(0x3 << 4);
   CCP1CON |= (pwm60_lower << 4);
   __delay_ms(3000);

   /********** 90% duty cycle **********/
   CCPR1L = pwm90_upper;
   CCP1CON &= ~(0x3 << 4);
   CCP1CON |= (pwm90_lower << 4);
   __delay_ms(3000);
}
```