```
// STUDENT NAME: Paul Martin
// ME4640 Robot Homework A - Servo Calibration Program
// This program allows you to control a servo motor position using serial input
// and measure the corresponding angles for calibration.

#include <Servo.h>

// ===== SERVO CONFIGURATION =====
Servo servo_1;
const int SERVO_PIN = 3;       // PWM pin for servo (can use 3, 9, 10, or 11)
const int MIN_PULSE = 579;     // Minimum pulse width in microseconds (you should adjust this)
const int MAX_PULSE = 2560;    // Maximum pulse width in microseconds (you should adjust this)
const int START_POSITION = 1570; // Starting position (approximately center) (You should adjust this for each servo)

// ===== CALIBRATION COEFFICIENTS =====
// TODO: After collecting data and performing curve fitting, update these values
// For linear fit: angle = SLOPE * microseconds + INTERCEPT
// For quadratic fit: angle = COEF_A * microseconds^2 + COEF_B * microseconds + COEF_C
const float SLOPE = 0.0909;     // Replace with your fitted slope (deg/µs)
const float INTERCEPT = -52.7;  // Replace with your fitted intercept (deg)

// ===== PROGRAM VARIABLES =====
int current_position_us = START_POSITION;  // Current commanded position in microseconds
int incomingByte = 0;                  // Storage for serial input
float calculated_angle = 0.0;          // Calculated angle from calibration equation
```

```
void setup() {
  // Initialize serial communication
  Serial.begin(9600);
  delay(2000);

  // Attach servo to pin with pulse width limits
  servo_1.attach(SERVO_PIN, MIN_PULSE, MAX_PULSE);

  // Move to starting position
  servo_1.writeMicroseconds(current_position_us);

  // Print instructions
  Serial.println("========================================");
  Serial.println("Servo Calibration Program");
  Serial.println("========================================");
  Serial.println("Controls:");
  Serial.println("  1: Decrease 100 microseconds");
  Serial.println("  2: Decrease 10 microseconds");
  Serial.println("  3: Decrease 1 microsecond");
  Serial.println("  4: Increase 1 microsecond");
  Serial.println("  5: Increase 10 microseconds");
  Serial.println("  6: Increase 100 microseconds");
  Serial.println("========================================");
  Serial.println("Input\tMicroseconds\tAngle(deg)");
  Serial.println("========================================");

  delay(1000); // Wait for servo to reach starting position
```

```
}

void loop() {
  // Check if serial data is available
  if (Serial.available() > 0) {
    incomingByte = Serial.read();

    // Process input using character literals (more readable than ASCII codes)
    switch(incomingByte) {
      case '1':
        current_position_us -= 200;
        break;
      case '2':
        current_position_us -= 10;
        break;
      case '3':
        current_position_us -= 1;
        break;
      case '4':
        current_position_us += 1;
        break;
      case '5':
        current_position_us += 10;
        break;
      case '6':
        current_position_us += 200;
        break;
      default:
```

```
    // Ignore invalid inputs
    return;
  }

  // Constrain position to valid range
  current_position_us = constrain(current_position_us, MIN_PULSE, MAX_PULSE);

  // Command the servo
  servo_1.writeMicroseconds(current_position_us);

  // Calculate angle using calibration equation
  // TODO: Update this equation based on your curve fitting results
  calculated_angle = SLOPE * current_position_us + INTERCEPT;

  // Print data in tab-separated format for easy copying to Excel/MATLAB
  Serial.print(incomingByte);
  Serial.print("\t");
  Serial.print(current_position_us);
  Serial.print("\t");
  Serial.print(calculated_angle, 2); // Print with 2 decimal places
  Serial.println();
 }

 delay(10); // Small delay to prevent serial buffer overflow
}
```