Spring 2021 – ECE 487/587 Lab #3 Grading Sheet

Name: _____ CWID: _____

Functionality/description of code (70 points):

Good Programming Practices (30 points):

1) Comments

2) Indentation

3) Good/meaningful variable names

4) Minimum/wise usage of global variables; unnecessary use of variables

5) Inefficient code

```
/*******************************************************************************
 * lab_3
 * Created by Matt Mason,
 * CWID 11800439
*******************************************************************************/

#include <avr/wdt.h>

// seven-segment display pins
#define SEVEN_SEG PORTC
#define SEVEN_SEG_DDR DDRC

// special characters
#define WAIT 16
#define ERROR 17
#define DP_ON 18
#define DP_OFF 19

/*******************************************************************************
 * Function:        updateDisplay
 * Parameters:      int character – the desired character from the characters
 *                  array to display
 * Return value:    none
 * Purpose:         Display a character on the seven-segment display wired to
 *                  SEVEN_SEG register pins.
*******************************************************************************/
void updateDisplay(const int character)
{
    // seven-segment display character mappings
    static constexpr byte characters[] = {
        B00000011, // 0
        B10011111, // 1
        B00100101, // 2
        B00001101, // 3
        B10011001, // 4
        B01001001, // 5
        B01000001, // 6
        B00011111, // 7
        B00000001, // 8
        B00001001, // 9
        B00010000, // A.
        B11000000, // b.
        B01100010, // C.
        B10000100, // d.
        B01100000, // E.
        B01110000, // F.
        B11111101, // WAIT
        B01101101, // ERROR
        B11111110, // DP_ON
        B11111111, // DP_OFF
    };
    // write charater pin values
    SEVEN_SEG = characters[character];
}

/*******************************************************************************
 * Function:        timeout
 * Parameters:      unsigned long startTime – time of last input in ms
 * Return value:    bool – true if 4 seconds have elapsed since startTime,
 *                  false otherwise
 * Purpose:         Check to see if the 4-second user input timer has expired.
*******************************************************************************/
bool timeout(const unsigned long startTime)
{
    return millis() > (startTime + 4000);
}

```

```
/*******************************************************************************
 * Function:        parseInput
 * Parameters:      String& input — user input string
 * Return value:    int — user input value 0–15, or -1 if input invalid
 * Purpose:         Parse user input and convert it to an integer, checking
 *                  for invalid input.
*******************************************************************************/
int parseInput(String& input)
{
    // trim any leading/trailing whitespace
    input.trim();
    // make sure string is not empty
    if (input.length() == 0)
        return -1;
    // make sure all remaining characters are numbers
    for (int i = 0; i < input.length(); i++)
        if (!isDigit(input[i]))
            return -1;
    // convert to integer
    int value = input.toInt();
    // return the value if it is in the allowed range 0–15
    if (value >= 0 && value <= 15)
        return value;
    // out of allowed range, invalid input
    return -1;
}

void setup()
{
    // configure all seven-seg. pins as outputs
    SEVEN_SEG_DDR = 0xFF;

    // open serial connection
    Serial.begin(9600);
    Serial.print("lab_3 by Matt Mason");
}

```

```
void loop()
{
    // prompt for user input
    updateDisplay(WAIT);
    Serial.print("\nPlease enter an integer between 0-15: ");
    // save the current time
    unsigned long startTime = millis();

    // continuously prompt for user input, but break if the user takes
    // longer than 4 seconds to respond
    String input = "";
    while (!timeout(startTime))
    {
        // read serial until buffer into input string until buffer is empty,
        // 4-second timeout expires, or newline received
        char c = 0;
        while (Serial.available() && !timeout(startTime))
        {
            c = (char)Serial.read();
            if (c == '\n')
                break;
            input += c;
        }
        // if last character reveived was a newline, user input is ready
        if (c == '\n')
        {
            // parse user input
            int value = parseInput(input);
            if (value == -1)
            {
                // show error message for invalid input
                updateDisplay(ERROR);
                Serial.print("Invalid input!");
            }
            else
            {
                // show input on seven-segment display
                updateDisplay(value);
                // print input and reaction time to serial monitor
                unsigned long reactionTime = millis() - startTime;
                Serial.println(value, HEX);
                Serial.print("reaction time = ");
                Serial.print(reactionTime / 1000.0f, 3);
            }
            // clear input string
            input = "";
            // prompt for user input
            Serial.print("\nPlease enter an integer between 0-15: ");
            // refresh startTime
            startTime = millis();
        }
    }

    // user took longer than 4 seconds to respond,
    // print a newline to simulate no input
    Serial.println();
    // blink decimal point for 4 seconds
    for (int i = 0; i < 4; i++)
    {
        updateDisplay(DP_ON);
        delay(500);
        updateDisplay(DP_OFF);
        delay(500);
    }
    // blink for 1 more second, using system watchdog to reset board afterwards
    updateDisplay(DP_ON);
    delay(500);
    updateDisplay(DP_OFF);
    wdt_enable(WDTO_500MS);
    // spin until board resets
    while (true);
}
```