ECE 2534 – Homework 5 (55 points)
A/D Conversion, Interrupt Service Routines, Glyphs and OLED Graphics

For each problem, write the code that solves the problem in its own .c file. Name your files accordingly: hw5_problem2, *etc*. Give consistent names to any header files you produce for the purpose of completing this assignment.

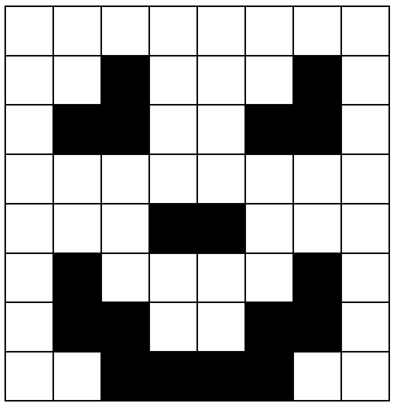
Submit a document containing your answers to Problem 1 and a zipped archive containing the source code for Problem 1b and 2. Use this naming convention for the archive that you submit:

<Last name>_<First name>_HW5.zip

Replace <Last name> and <First name> by your family name and given name respectively.

Problem 1 (15 points)

a. Imagine that the "icon" shown below is to be mapped to the OLED as a custom character. Follow the conventions described in ChrFont0.c to create the array that would display the icon as a character on the OLED.



BYTE problem1a[8] = {______, ______, ______, ______, ______, ______, ______, ______}

b. Design your own jack o'lantern face using the array below. In the space below, write down the BYTE array that defines your face. Modify the "Glyph_Ex_main.c", available in the Scholar site's Resources / Example code folder, to draw your face, and verify that it is correct. Submit your program with your HW solutions.

BYTE problem1b[8] = {______, ______, ______, ______, ______, ______, ______, ______}

c. Imagine that the character array shown below represents an OLED custom character. Follow the conventions described in ChrFont0.c to show what the icon would look like when displayed on the OLED.

BYTE problem1c[8] = {0xFF, 0x18, 0x18, 0xFF, 0x00, 0xFA, 0x1C, 0xBF}

<u>Problem 2</u> (40 points)
Write a C routine to perform the two interrupt-driven tasks described below. Your OLED will serve as a display for the tasks that your routine is performing.

```
L/R = XXXX  ZZZZ
U/D = YYYY
LEFT       RIGHT
UP          DOWN
```

The values XXXX and YYYY represent the digital values that the Cerebot analog-to-digital converter interprets from the analog outputs of your joystick. The two values correspond to the left-right and up-down axes, respectively. Since the ADC on the Cerebot is a 10-bit converter, the digital values that it can produce nominally range from 0 to 1023. You may find that your joystick does not deliver the full scale of analog voltages to produce the corresponding scale of digital values; this is okay.

Your Cerebot obtains the digital values by sampling the channels that you will connect to the L/R and U/D pins of your joystick. Your ADC configuration should be written in a way that collects the two samples from the channels in question, **then requests an interrupt** to read and process the values. Your **ISR** should read the values, assign them to variables, and use the values to determine how the joystick is being manipulated. The words "LEFT", "RIGHT", "UP", and "DOWN" should only appear on the OLED when the joystick is being moved in one of those directions. Since the two axes report values independently, it is possible for the joystick to be "LEFT" and "DOWN" at the same time, or "RIGHT" and "DOWN", and so forth.

Nominally the following should be true:
- The digital values for both axes should be approximately 511 when the joystick is at its neutral position.
- LEFT and DOWN correspond to digital values of 0 on their respective axes.
- UP and RIGHT correspond to 1023 on their respective axes.

Since potentiometers vary, the actual values could be off. Additionally, noise and vibrations might cause the digital values to "wobble" about their center point values. To account for this, you should use your ISR to determine whether or not the joystick is in a "neutral zone" around the center point. Only when the digital value is sufficiently greater than or less than the center value should your OLED report that the joystick is in one or more of the four displayable positions. You should choose a value that gives you the kind of joystick control you might want in a game that involves the Cerebot; experiment with values that produce a "neutral zone" around the center point that is to your liking.

The value ZZZZ is the value of a second counter of the sort that you wrote in the previous homework assignment. Even though you could incorporate the code that uses TMR23 directly into the grand interrupt configuration routine and interrupt service routine that you write, *you should consider returning to using a 16-bit timer with a small time base – like one millisecond – and using the millisecond real-time interrupt interval to derive the value of a higher time increment – like one second.* (As you saw in the previous homework, the rationale is that it is simpler to use a smaller base unit to time many events whose periods can be subdivided into the smaller unit than it is to time many events whose periods must be subdivisions of a larger time base.)

You should configure your Cerebot to use vectored interrupts; thus, you should have separate ISRs for each of the two possible interrupt sources (ADC or Timer).

Your main routine should configure the timer, the ADC, the OLED, and the interrupt controller. In an infinite-WHILE loop, it should update the OLED based upon the information that is updated by the occurrence of interrupts.

Here are some sample pictures to give you an idea of how your OLED should look in different situations: