CSE 102 – Assignment 1

Introduction to Assignments

What we are calling "Assignments" in CSE 102 are essentially what are called "Labs" in CSE 131. The "Assignment" label is simply to start moving away from the usage of the word "lab" to mean multiple things (e.g., workspace vs. assignment).

For this assignment, which builds off the metronome studio exercise, you are welcome to collaborate with folks up to getting the studio metronome functioning to your liking. Once you are to that point, however, this is a solo assignment, to be completed by yourself. You are free to ask for help from the instructors and/or TA, but don't share your solution with your peers (or ask for theirs).

Metronome

This assignment will build off the metronome portion of Studio 1, so if you haven't yet completed that portion of Studio 1, you will want to do that first.

From the studio, you have two constants that control the metronome:

```
    const int LEDOnTime_ms = <n>;
    const int metronomeRate_bpm = <b>;
```

For the assignment, you will add some additional capability to the metronome, so that it presents rhythms. We will call the assignment `rhythms`.

Expand the Digital Outputs

The initial task for the assignment is to add 3 more digital output LEDs (for a total of 4). Reference the Tutorial on Arrays to see how to use an array to hold pin numbers. (You do not need to use the same pins as in the tutorial.) Use 330Ω resistors in series with each LED.

Adapt the code from the tutorial (or develop your own approach) to ensure that your LEDs are wired correctly.

Specifying the Rhythm

You will be expanding your earlier `metronome` to now present `rhythms`. Each LED output will have a repeating rhythmic pattern, and the pattern will be specified via a `pattern` array declared as a two-dimensional array of `const int`:

```
    const int nChannels = 4;
    const int nMeasures = <m>;
```

```
const int pattern[nChannels][nMeasures] = {
     { … }, { … }, { … }, { … }
};
```

The entries in the pattern array indicate the number of (equally spaced) flashes each LED should have in each measure.

You should also specify the number of beats per measure:

```
const int beatsPerMeasure = <B>;
```

As an example, consider a case where `beatsPerMeasure = 4` (think of this as a 4/4 time signature for those who are musically inclined) and `nMeasures = 3`. We are therefore specifying what happens for 12 beats. If the pattern array for channel `0` is `{2, 4, 1}`, then in the first measure there are two flashes (in beats 1 and 3 out of 4), in the second measure there are four flashes (every beat has a flash), and in the third measure there is only one flash (on beat 1 out of 4).

For the musically inclined, we just described two half notes, four quarter notes, and one whole note. For those who are not familiar with musical notation, you don't need to know this.

In combination with the parameters from the metronome, you should now have a complete specification for the rhythmic patterns to be produced.

Write the Software

Given the specification above, author the `rhythms` code to implement the flashing LEDs. Make sure you use the free-running timer, millis(), to keep track of the passage of time.

Show your running `rhythms` to an instructor or TA, changing the specification several times to illustrate its operation.

Note that the specification that we are following is pretty restrictive. For example, each channel can only do one thing in each measure (e.g., we can't specify to flash on beats 1, 3, and 4 of a measure). Can you propose an alternative specification that is more robust? Or can you figure out how to address the above example limitation within the existing specification?

Explore its Capabilities

Investigate the range of effective parameter inputs that work well. What are the upper and/or lower limits to the parameters, and what causes those limits? E.g., can I specify triplet sixteenth notes?

Submitting Your Work

When you have finished your assignment and demonstrated it to the instructor or TA, make sure they record your completion.