**CMPUT 274/5 - Tangible Computing** **Fall 2019**
**Weekly Exercise #6**

---

# Counting Lights

---

This exercise is based on Lab 5: Counting Lights in the Arduino Intro Labs.

**Important Notes: Read This!**

1. There is a quiz component for this exercise that can be found on eClass above the code submission link. Both the quiz and code submission are required for full marks.

2. Your code must be a .cpp file that is compiled with the Makefile from the terminal. It must contain a main() function. We will not load it into the Arduino IDE!

3. No template will be provided for this exercise. Part of your task is to learn what you must include in a C++ Arduino file in order for it to run successfully.

4. You are expected to follow a variation of the wiring from the full Tug of War lab, (using the full five LEDs from the *Your Task* section rather than 3). To be clear, we expect you to use the following pins *exactly*:

   - Use pins 9,10,11,12,13 for the LEDs, such that pin 11 corresponds to the green LED in the center. The "first LED" should be the one connected to pin 9.
   - Use pin 6 for the increment button (for **Part 1**).
   - Use pin 7 for the decrement button (for **Part 2**).
   - Use pin 5 for the store/recall button (for **Part 3**).

   Marks will be deducted if you change the pins used for the lights or buttons. Your grading TA expects not to have to change the wiring for every submission! Please be kind to them by following these specifications.

## Part 1: Increment Button

Implement the desired behaviour outlined in Section 6.2 of the Arduino Intro Labs. Specifically, you are asked to create the functionality of the increment button. Please make sure to read section 6.3 before getting started: for full marks you will need to implement this using loops.

**Read the intro labs carefully** to see how to properly wire and use a push button.

## Part 2: Decrement Button

Modify your program so that the second pushbutton performs the decrement operation mentioned in Thought Question 4. This must also be done using loops. If you decrement when all LEDs are off, it should turn them all on (so this button "reverses" the action of the increment button).

**Note:** For full marks, your program must increment or decrement exactly once each time the corresponding button is pressed down. That is, holding the button down for a while should not cause multiple increments or decrements.

## Part 3: Memory Button

Finally, add a third pushbutton for a store/recall feature. The first time this button is pushed, the value of the 5 LEDs should be stored in another "memory" array and then all LEDs should be reset to OFF. The second time the button is pushed, the LEDs should again be lit up in the same way as when you first pushed the store/recall feature (i.e. the stored states of the LEDs is recalled).

The third push should act like the first push: store the current state of the LEDs. In general, the effect of pushing the button should alternate between storing the values of the LEDs when the button is pushed, and recalling the values of the LEDs from last time it was pushed.

We should be able to interleave increment, decrement, and store/recall operations. That is, after we store the LEDs and they are reset, the increment button should just turn the first LED on. After we recall the LEDs, the increment/decrement buttons should affect the sequence that was just recalled.

## Submission Guidelines:

Submit all of the following files in a .tar.gz called **counting.tar.gz** or a .zip called **counting.zip**:

- counting.cpp, containing your implementation of the weekly exercise.
- the Makefile (introduced in class and available on the VM)
- your `README`, following the Code Submission Guidelines for Arduino projects. **Reread the guidelines so you are familiar with the expectations for an Arduino project**.

Note that your files must be named **exactly** as specified above.

## Loops and Arrays

Part of the purpose of this exercise is to force you to use arrays and loops properly. So, the current state of the LEDs must be stored in an array, just like in the intro lab and you should use loops to perform the increment and decrement (and anywhere else you need). In particular, we are aware that there is a cute solution using bitwise operators (not yet covered in class) and integer addition to perform the increment. But you have to use an array for this exercise.

**Style**

Your submission must use good style. Generally speaking, good style rules from Python that are applicable to C++ still apply: comments must be placed appropriately, variable names must make sense, you should exercise good use of functions, etc. With C++, you also **must have proper indenting**. While a C++ program with bad indenting can still be compiled, you must indent properly.

Finally, your program must have a `main()` function that calls a `setup()` function to initialize things like the Arduino, the pin modes, and anything else that is appropriate here. Note, Serial communication is not required for this weekly exercise but you will not be penalized for initializing it anyway.