Laboratory 2– Expressions and selection

Objectives: This laboratory is focused on expressions and selection (conditional) statements in C

Learning outcomes:

- Wrap an incrementing counter to a range
- Use basic, nested, and cascaded if statements

Lab instructions

1. The lab is for individuals and must be completed during the lab session
2. Create a new sketch for each major section in the lab.
3. Before you leave the lab, call the lab demonstrator to check what you have done for all the sections (this is why separate sketches are important). *Anything not checked by the demonstrator during lab will have to be assigned zero marks.*
4. Create a single plain text submission file (.txt) for the lab. Use a plain text editor (e.g. NotePad++) to edit the .txt file and do not use Microsoft word or similar. Copy all the sketches you write and any other answers required for the lab into the submission text file. *Name the file "108_Lab2_firstname_surname.txt". Include your name and lab number at the top of the file and clearly label everything in the file. Submissions without names or unclear sections/sketches/answers will be marked down or (in the worst case) not marked at all.*

Marking for lab/assignment

The lab is marked during the lab session based on demonstrated behaviour and brief code inspection. It is essential that you get both the running sketch and code checked during the lab session.

For all code sections, marks will be deducted for bad communication and style (e.g. missing or mismatching comments, poor variable names, bad indentation, etc.), incorrect behaviour, or failure to follow requirements of the lab question.

General marks are also lost if the submission document instructions are not followed.

1 Nested if-statements — save sketch as “Lab2_1_NestedIf”

(5 marks)

Background: Read the question to the end, before starting. You will be writing a short sketch which loops every 100 ms checking SW1 and then lights LED1 or LED2 depending on how long SW1 has been pressed.

Lab2_1_NestedIf requirements:

- You must solve this problem using a nested if-statement
- The loop function must repeat once every 100 ms, so add an appropriate delay at the end of the function but ensure it does not delay longer than this.
- When the button (also called a momentary switch) SW1 is pressed, light LED1.
- If SW1 is pressed each consecutive time you check for 1.5 secs or longer, light LED2 in addition to LED1.
- If SW1 is not currently pressed both LED1 and LED2 must be off.

Hints: you should check the button on each loop and use a static local counter variable to keep track of how many consecutive ticks (repetitions of the loop function) the button has been pressed for. Define a constant to be the number of ticks that corresponds to 1.5 secs and then check to see if you equal or exceed that number of ticks.

Don’t forget to set the pinMode for your LEDs in the setup function.

To check if SW1 is pressed you need two things. The switches on the daughterboard are Active-Low, which means that they read as logic HIGH when not pressed and LOW when pressed. Because this is a bit counter-intuitive we have defined a constant, SW_ACTIVE, in the ee108 library that you should use.

Now, when we want to check if the button SW1 is pressed we can write

```
            if (digitalRead(SW1_PIN) == SW_ACTIVE) …
```

Finally, don’t forget about comments, naming of variables and constants, etc. if you want full marks.

✏ Copy the sketch into your answer document and ensure you demonstrate it also.

2 Tutorial – wrap or reset a counter variable at a specific value — save sketch as “Lab2_2_WrapTutorial”

(No marks)

There are two common ways to reset or wrap an incrementing counter variable at a specific value. Assuming we’ve defined a constant, WRAP_VALUE, which specifies the value at this to wrap, you can use an if-statement as follows:

```
++counter;
if (counter >= WRAP_VALUE)
    counter = 0;
```

Another strategy is to use the mod operator to wrap at the specified value as follows:

```
counter = (counter + 1) % WRAP_VALUE;
```

Why does this work? Remember that x % n is the remainder after integer division x/n. Therefore the remainder can only ever be a number between 0 and n-1 inclusive. A general idiom is therefore

```
limitedRangeNumber = largeNumber % RANGE_LIMIT
```

Download the starter sketch of moodle and extract the zip file inside your sketchbook folder. The sketch should now appear in the Arduino IDE under File > Sketchbook > Lab2 > Lab2_1_TutorialStarter.

At the comment marked “TODO” add in one of the methods above for incrementing and wrapping the counter. Satisfy yourself that it is working correctly. Then comment it out (if you just place your cursor on the line and press **`Ctrl`** + **`/`** you can comment (or uncomment) the line). Now try the other method for wrapping the counter and satisfy yourself that it works.

Reminder: don’t forget to put a comment at the top of each sketch as in lab 1. Do this for every sketch in the lab.

✏ Copy the sketch into your answer document. You do not need to demonstrate it.

3 Cascaded if-statements — save sketch as "Lab2_3_CascadedIf"

(5 marks)

Background: Read the question to the end, before starting. You'll be writing a short programme which increments a counter once per loop and lights a particular pattern of LED1 and LED2 according to the counter value.

Lab2_3_CascadedIf requirements:

- You must use a cascaded if-statement to solve this lab problem
- Allow the loop function to repeat once every second (delay at the end of the loop)
- Implement a counter which counts up by 1 on each repetition of the loop function. It should wrap when necessary so that it is restricted to the range 0 to 9 inclusive (see Lab2_2_WrapTutorial).
- On each repetition of the loop print the counter value and a terse description of what the LED status should be (as described in the following bullets). The output should appear like:

```
…
counter = 1, Both LEDs off
counter = 2, LED1 only
counter = 3, LED2 only
…
```

- When the counter is between 0 and 1 or 8 and 9 inclusive, both LEDs should be off
- When the counter is 2 or 7, only LED1 should be on
- When the counter is 3 or 6, only LED2 should be on
- When the counter is 4 or 5, both LEDs should be on

Hints: Don't forget to initialize pinMode and serial output in your setup function (see Lab1).

You will probably need to use logical expressions (with && or ||) in your if-statements for the conditions above.

The last piece of the cascaded-if statement should be an "else" that handles all conditions you have not explicitly checked for previously.

✏ Copy the sketch into your answer document and ensure you demonstrate it also.