Objectives: This laboratory aims to practice using scalar pointers and more arrays in C on Arduino.

Learning outcomes:

- Declare and use scalar pointers including the address-of and dereference operators
- Declare and use functions that have output or input/output parameters based on pointers
- Use an array as a first-in-first-out buffer

Lab instructions

1. The lab is for <u>individuals</u> working alone 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 <u>single plain text submission file (.txt)</u> for the lab using a plain text editor (e.g. NotePad++). Copy all the sketches you write and any other answers required for the lab into the submission text file. *Name the file "108_Lab6_firstname_surname.txt", using your actual name. Include your name and lab number at the top of the submission file also and clearly label everything in the file. Unclear submissions will have to be marked down or (in the worst case) not marked at all.*

Marking for lab/assignment

Most of the lab and assignment will be marked during lab sessions. It is essential that you get the demonstrator to confirm your progress during the lab (for the lab portion) and at the start of the next lab (for the assignment portion).

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

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

1   Function output parameters –
save sketch as "Lab6_RectOutputParams"

Background: review notes 2.20 Scalar Pointers.

The objective of this first exercise is to practice using pointers, the address of operator and the dereference operator in the context of a function with multiple output parameters. The function in question is very simple and just calculates the area and perimeter of a rectangle given its length and width.

Lab6_RectOutputParams requirements:

- Download the Lab6_RectOutputParamsStarter sketch from moodle to get you started on this exercise and extract it to your sketchbook folder.
- Define a function called **rectangle** which takes 2 input (i.e. normal) parameters and 2 output parameters (using pointers). The input parameters are the length and width of the rectangle (both of int type). The two output parameters are the area and perimeter of the rectangle respectively as calculated by the function. Both of these parameters must be int pointers.
- In your loop function declare 4 ordinary <u>local</u> variables: 2 to hold the length and width and 2 to receive the output from the rectangle function.
- When the SW1 button is clicked:
    ○ Generate random values (in the range 0-99 inclusive) for length and width
    ○ Call the rectangle function you wrote to calculate area and perimeter based on the length and width. Pass pointers to the variables which will receive the function outputs.
    ○ Print all 4 values to serial out
- Your programme should generate output something like the following:

```
Lab6_OutputParams starting...
Press SW1 to start

Rectangle: length 7, width 49 => area 343, perimeter 112

Press SW1 to do another rectangle
```

🖉 Copy the sketch into your answer document and demonstrate it.

2   In-out and optional output parameters –
save sketch as "Lab6_AnalogInoutAndFifo"

Background: The objective of this exercise is to use an array as a first-in-first-out (FIFO) buffer and to define and use a function with optional inout parameters. The programme will read analog input from the light dependent resistor on the EE108 board, optionally determine the min and max value seen to date, and then add the analog reading to a FIFO buffer for the purpose of calculating an average of the recently seen analog readings.

Lab6_AnalogInoutAndFifo requirements:

- Download the starter sketch, "Lab6_AnalogInoutAndFifo", from the moodle lab downloads and extract it into your sketchbook folder.

    ○ Save it as the appropriate sketch name for your submission.
    ○ The starter code reads analog input from the light dependent resistor (LDR) so make sure that you set the jumper J2 to select the LDR on the EE108 daughterboard.

- Modify the sampleAnalogIn function so that it can optionally output the min and max sample values seen to date. To do this:

    ○ Add two in-out parameters, one for the min and one for the max. The caller can supply the current min and max values seen and the sampleAnalogIn function will then update these if the current sample value is bigger than max or less than min.

    *IMPORTANT: Don't use the names **min** or **max** for variables or parameters because these clash with macro names in the standard library and strange errors will result.*

    ○ Make the min and max parameters optional by adding code in sampleAnalogIn to check if the parameters are valid (i.e. that the pointer is not NULL) before using it. If the pointer is NULL, the pointer must not be dereferenced (either to read the input value or to set the output value).
    ○ Modify the function declaration (near the top of the file) to match any changes to the parameter list in the function definition.
    ○ Modify any existing calls to the sampleAnalogIn function (from the starter code) so that the function call matches the function definition/declaration but <u>do not</u> use the optional in-out parameters. In C you cannot simply not provide an argument – instead you must pass in NULL to make it clear you don't want to use this optional parameter.
    ○ If you run the code at this point, the visible behaviour should be identical to the starter code.

- Now uncomment the block in the starter code which relates to sampling <u>with</u> the optional parameters. Modify the call to sampleAnalogIn so that it updates the min and max values each time it is called by setting the inout parameters to valid pointers. Print the sample value , min, and max after each call.

    Hint: to ensure that the variables you use for min and max start from reasonable defaults it is traditional to initialise the max to the smallest value possible and initialise the min to the largest value possible. (That way, the first value read will cause the min and max to be updated.) For integers in C the smallest and largest int values are defined in the include file "limits.h" as INT_MIN and INT_MAX.

- The output should look like:

```
-----------
Sampling with no optional params...
Sample 1, value=969
Sample 2, value=973
…

<continued on next page>
```

```
sampling with optional params...
Sample 1, value=971, min=971, max=971
Sample 2, value=954, min=954, max=971
…
```

- Finally add support for a first-in-first-out (FIFO) buffer that can be used to calculate the average of the most recent samples:
    - First declare an array of size 4 to act as the FIFO. It must remember its values between calls to the loop function.
    - Now define a function called addToFifoAndAverage. This will take the analog reading and the FIFO buffer (and length) as parameters and will return the average of the elements in the FIFO. Before calculating the average, the function must copy all elements of the array up one place and then overwrite the first element with the latest analog reading, e.g. if the FIFO initially contains { 4, 3, 2, 1 }, then after adding new reading 5 it should contain { 5, 4, 3, 2 }
    - Modify the loop function to call addToFifoAndAverage after calling sampleAnalogIn (with the optional parameters) and update the printing to include the FIFO contents and average of the FIFO values. The output should look like

```
-----------
Sampling with no optional params...
Sample 1, value=969
Sample 2, value=973
…

sampling with optional params...
Sample 1, value=933, min=933, max=933, Fifo = {933, 0, 0, 0}, average = 0
Sample 2, value=930, min=930, max=933, Fifo = {930, 933, 0, 0}, average = 233
Sample 3, value=944, min=930, max=944, Fifo = {944, 930, 933, 0}, average = 465
Sample 4, value=935, min=930, max=944, Fifo = {935, 944, 930, 933}, average = 701
Sample 5, value=928, min=928, max=944, Fifo = {928, 935, 944, 930}, average = 702
…
```

✎ Copy the sketch into your answer document and demonstrate it.