Objectives: This laboratory aims to practice writing and using functions in C.

Learning outcomes: Declare, define and use (call) basic functions

---

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_Lab4_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 Practice using "pure" functions that return a value – save sketch as "Lab4_PureFunctions"

Background: Review notes 1.50 and "pure" functions in particular. Pure functions always return a value and do not have side effects. They are most similar to the mathematical functions.

The purpose of this part of the lab is to define a new "pure" function and then gain experience with the various ways such a function can be used, including to assign the return value to variable, to call the function as part of an expression, to call the function as an argument to another function, etc.

You should particularly refer to later parts of notes 1.50 from slide 32 onwards. These explain how a function can be called as part of an expression or as arguments to a function call and also describe how to declare a function. See also the downloadable example **Lec150_SimplePureFunction** under example sketches for functions on moodle.

Lab4_PureFunctions sketch requirements:

- Declare and define a new function to calculate the absolute difference of two integers. The resulting absolute difference is also an integer. The definition of the absolute difference is that it must always be positive, regardless of sign of the two numbers and regardless of whether the first or second number is bigger. E.g. the difference of 5 and 15 is 5-15 = -10 but the absolute difference is 10. Mathematically, this would be defined as:

$$f(x, y) := |x - y|$$

- Ensure that you give your function a sensible name and that you write the <u>function declaration</u> (or prototype) above/before the `setup` and `loop` functions, but write the <u>function definition</u> (which includes the function body code) below/after the `setup` and `loop` functions.

- Now, execute each of the following in the order specified once every time the loop function runs.

  ○ First generate 4 random integers, *p*, *q*, *r*, and *s,* each between 0 and 100 and print each one. (Make sure we can see which is which.) Use `random(100)` to generate each number, e.g. `p = random(100);`

  ○ **Use in assignment:** Use your function to get the absolute difference of *q* and *r* and assign the return value to the variable `result`. Print the result to serial.

  ○ **Use directly in an expression:** Test if the absolute difference of *p* and *r* is greater than *q* and print either "abs diff p and r > q is true" or "abs diff p and r > q is false" as appropriate. <u>Important:</u> call your absolute difference function directly in the conditional expression to avoid using unnecessary temporary variables.

Question continues on next page…

- ○ **Using a function directly as an argument to another function call:** <u>In a single line of code</u>, calculate the absolute difference between the absolute difference of *p* and *q* and the absolute difference of *r* and *s*. Print the result.

   Hint: it is perfectly legal to use a function call as an argument to a function similar to how you might in a mathematical expression like: y = f(f(a,d), f(b,c))

- Each run through the loop function should produce well formatted output like this:

```
--------
p is 10, q is 20, r is 15, s is 75
abs diff q and r is 5
abs diff p and r > q is false
abs diff of abs diff p and q and abs diff r and s is 50
…
```

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

2   Practice using functions with side effects
– save sketch as "Lab4_BlinkFunctions"

Background: This part of the lab focuses on functions that have side-effects (in contrast to the "pure" functions in part 1).

Your task is to *refactor* the sketch supplied with the lab to minimize the amount of code repetition by using functions. *Refactoring* means keeping the exact same externally visible behaviour while changing the way the code is implemented.

Download the sketch **Lab4_BlinkNoFunctions** supplied with your lab and extract it into your sketchbook (if you have not already done so). You should then be able to find it in File > sketchbook > Lab4_Resources > Lab4_BlinkNoFunctions.

The supplied sketch already implements the following requirements without using any custom functions:

- Sets up the sketch for serial output and use of the BarLEDs
- Each time the loop function is executed it does the following

  - ○ Blink BarLED1 once using a long pattern (1200ms on, 600ms off)
  - ○ Blink BarLED8 three times using a short pattern (100ms on, 500ms off)
  - ○ Blink BarLED4 twice using a medium pattern (500ms on, 700ms off)

- It prints output similar to the following when it runs

```
Loop 0
  Blink BarLED1 on/off long pattern x 1
  Blink BarLED8 on/off short pattern x 3 times
  Blink BarLED8 on/off short pattern x 2 times
Loop 1
  Blink BarLED1 on/off long pattern x 1
…
```

If you examine the code for Lab4_BlinkNoFunctions it should be clear that it is quite repetitive, particularly with regard to blinking the LEDs. The same few lines of code are used each time we

want to blink the LEDs (although some specific details vary, such as the LED to blink, the on and off duration, and the number of times to blink the LED).

Lab4_BlinkFunctions sketch requirements:

- Download **Lab4_BlinkNoFunctions** to use as your starting point if you have not already done so (see above).
- Declare and define a new function called `setupBarLeds`. The purpose of this new function is to move the couple of lines of code related to setting up the BarLED pins for output into their own function. Clearly, once you have implemented the `setupBarLeds` function, you should modify the setup function to replace the original couple of lines that set up the bar LEDs with a call to the function `setupBarLeds`.
- Declare and define a new function that can blink any BarLED number you specify some specified number of times and with specified blink on/off durations. Now replace all the existing repetitive blink code in the loop function with calls to this new function instead. Your objective is to minimize the amount of repeated code.

Hints: Give the function you create a sensible, self-explanatory name. The led, the blink on and off durations in ms, and the number of times to repeat the blink should all be parameters to the function. As all the bar LED pins are consecutive, it is possible to calculate the pin number of any bar LED number by addition of the appropriate offset to BAR_LED_1_PIN.

IMPORTANT: *although the lines of code to print some information to serial are repeated alongside the blink code in the loop function, printing to serial is a different (and unrelated) task to blinking an LED so <u>don't include printing inside your new function</u>. Instead try to make each function focus on doing one thing well – in this case, blinking the LED.*

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