Lab 0: Re-Introduction to C

The goal of this inital lab is to re-acquaint yourself with the `C` low-level, systems based nature of `C`.
The majority of the lab-work in this semester will be coded in `C` and so getting over the typical 'teething issues'
often encountered in such a language will make engaging with the actual lab-content easier and more useful.

Task 1: Opening a file

We have a text-file `textfile.txt`. We will be asking you to open the file and perform various actions with its
contents.
To make things easier, we have provided basic source file (task1.c) which (wrongly) attempts to open and read a file.

Easy - Fix the example

The provided source code attempts to open a text file and print out its contents. While all the required function calls
are present there are a number of basic errors.

Medium - Provide the filename

We would now like you to extend the example to take the name of a file to open as input.

Hard - Every second word

Now extend your code to print every second word in each line. We recommend using the function `strtok` to do this
safely.

Food for thought

- What happens after you are finished with a file?
- How large can each line be with your program? Is there a way to allocate a buffer to fit each line exactly?
- Could you adapt your code to be passed a filename as a command-line argument rather than having to input it at runtime?
- Try running `$ man strtok` to find out more about this function call.

Task 2: Compiling with Headers

While you may be used to writing your own code from scratch or dealing with simple examples, in the labs for this unit you will often need to adapt often extensive code, use many libraries and more than a single source file to solve your problems. We would like to get you used to this now. We have two files `checksum.h` and `checksum_ccitt.c` which implement some basic checksum functions for you. We woud like you to write another source file which implements the following tasks.

Easy - Checksum a string

Write a source file which runs the string 'hello world' through the `checksum_ccitt` function provided for you. Your solution should only require 4 lines of code to run perfectly; the main challenge here is to find out what arguments `checksum_ccitt` requires and how to compile with multiple source files

Medium - Checksum a file

We would now like you to adapt your solution to task 1 to print the checksum of each line in a file.

Hard - Checking a checksum

We now have two additional files `test.in` and `soln.out`. The former is simply a text file, the latter contains the checksum of each line. Please write some code which runs `checksum_ccitt` on each line found `input.in` and prints an error message if the result does not match the corresponding line in `checksum.out`.

Food for thought

- Can you automate the build process for multi-file projects. (Remember makefiles?)
- While not as much of an issue now, for many years different processors could represent numbers differently; ordering the bytes of integers in different ways. Network code needs to be able to deal with such differences termed the 'endianness' of a machine.