CSC9V4 Practical 1
Getting to Compilation

C is more than 40 years old. There are multiple compilers and IDEs. The aim of this practical is to execute a *code-compile-(fix code, if needed)-run* cycle, play with development environments and get used to one (or more) of them.

Very effective, learning-wise, is to use a combination of a simple text editor and terminal/command shell: programs are written and saved in the editor and the *gcc* invoked in the terminal - be careful to repeat the cycle write/save/compile/run all the time you change your program (check that the a.out/a.exe file has permissions for execution). You need to choose a suitable editor and install a version of gcc.

In the linux/mac world, the simplest thing is to use a suitable editor - a very effective one is emacs (old school!) - and a standard terminal.

In the Windows platform there are a few options (read them all, first):

The Netbeans (IDE) + MinGW (compiler package) is quite used - and it is what is normally installed in our labs on campus. **If** you want to install and use it on your computer, suggestions on how to set C as the default for new projects follow (see below for alternatives to Netbeans!):

- Launch a new Project
- Select C/C++ in the left pane; now C/C++ Application in the right pane; click Next.
- **Make sure:** To select C from drop-down. This should now be set as the default.
- **Make sure:** To change the *Location* from UNC path, i.e. \\... , to **H:\**. UNC is unsupported in Netbeans.
- Give your project a name, ie. myFirstC, and continue.
- NOTE: Standard practice is to create a new folder for each project, with the same name.

Open `main.c` by double-clicking in the Project pane (left) under ‘Source Files’.

Strongly Recommended: Set the `–Wall` compiler option:

- In the Project pane (left), right-click on the project name → Properties.
- Now Build → C Compiler → click ‘Additional Options’.
- Type `–Wall` and accept the change.

Getting Familiar: Now take the time to mouse-over and become familiar with the icons in the toolbar. Take special note of all buttons that compile and/or run. In the Project pane on the left, click the ‘+’ symbol to reveal all of the projects resources (source files, images, etc). There should appear only main.c -- Clearly, source files are edited in the large text editing pane in the centre of the screen.

Compile and run your program now. A CLI window should appear, even though it does nothing. The bottom window expands to reveal compiler and linker feedback, as well as errors and warnings if there are any. Feel free to navigate, investigate, and play around. The worst that can happen is a restart of the IDE!

Dev-C++ is a full-featured C and C++ Integrated Development Environment (IDE) for Windows platforms, quite widely used:

http://dev-cpp.com/

If you are "adventurous" and happy to experiment with new things, you can look at tools like:

Windows Subsystem for Linux (WSL)- a linux subsystem on windows core that provides you with the standard linux gcc suite, and you can also Nodepad++ that recognises C syntax

https://docs.microsoft.com/en-us/windows/wsl/install-win10
https://notepad-plus-plus.org/downloads/

Cygwin, another linux-like environment that can be installed on Windows

https://www.cygwin.com/

MS Visual Studio - quite powerful and the community edition is generally free

https://visualstudio.microsoft.com/

Another easy solution - which might work reasonably well for our purposes and for any platform, is to use an online gcc editor/compiler/executor, e.g.

https://www.tutorialspoint.com/compile_c_online.php

It is very straightforward to write simple programs in the editor on the left, compile/execute them by pressing the Execute button, and see the results of the computation on the right pane.

Note: Working remotely with a large number of different computers, OSs, installations ... it will be impractical to refer to a standard installation. It is *very important* that each of you sets up an environment with which they can work comfortably. Please, spend some time experimenting with the solutions above - at the minimum, the online solution should work for all in a first instance.

Do something!

Time to write code! Start with hello world. If choosing to use a text editor + command line instead of an IDE, remember that every C program must at a minimum:

- Include headers; all CLIs will need `stdio.h`
- The `main()` 'driver'. IDEs generate function arguments for program parameters. If none are required, then `void` is sufficient as a parameter.
- Finishing with `return 0 or equivalent;` is a good idea to let the system know that the program has completed without incident.

C syntax is very similar to Java. Even printf() is available in Java, though it is the convention in C. Taking as an example the following for-loop in C (similar to Java, isn't it? - what does %d mean?) try to write programs that compute

- the squares (x*x) from 1 to 10;
- the multiplication table from 1 to 10 (by using two nested for-loops).

```
for ( int x = 1; x <= 100; x++)
        {
        printf("%d\n", x);
        }
```