COMP281 Practical Sessions
Week Commencing 5th Feb 2024 (week 2)

This week in the practical sessions for COMP281, you're going to do some C programming using a standard graphical editor and the command line C compiler via a linux-like environment. You're also going to try to program the solution to some simple problems.

Instructions:

Using your own Mac: You'll need to install Xcode from Apple's App Store. It's a big download so it may take some time. Once you've downloaded it, run it (and agree to install the command-line tools). Once it's running, quit it. Open the Terminal App (you'll find it in the Utilities folder inside your Applications folder).

Using a CS lab or Rendall PC Teaching Centre PC: startup the Cygwin64 terminal from the desktop icon.

Using a different Uni lab PC: On these, Cygwin will not be installed and so we have to use the "install university applications" program. Start that program, from the list on the left choose "all", and then scroll the list on the right until you see "Cygwin compilers for CompSci 3.2.0" and highlight it. Click the install button and wait a while (it may take some time). You should find that there's now a Cygwin64 Terminal on your desktop. Start that up.

On your own PC: Download and install Cygwin and the gcc-core package from the Cygwin project homepage:

https://cygwin.com

Once it's installed, start it up.

Type `pwd` and press return. This will show you the current folder. If this is not M: (or something like that), then you will need to change it to that (M: represents your university home folder aka M "drive").

Cygwin is a sort of linux environment under Windows. By default Cygwin on a CS dept PC sets the working directory to your M "drive". If your current directory is not, then type

`cd M:` and press return (note that the ":" is important)

On your own PC, choose a good location for the folder that will contain your COMP281 program source files.

Make a COMP281 folder, and in the Cygwin terminal window (or the terminal on a Mac), cd to COMP281 (on a real linux system you'd have to use UPPERCASE where appropriate. On this Windows look-alike, filename case is apparently ignored as it normally is on Windows).

Using a CS lab PC: startup the Atom text editor from the desktop icon.

Using a non-CS Uni lab PC: On a University non-CS PC, we have to use the “install university applications” program, to install Atom. Scroll the list on the right until you see “Atom Text Editor 1.60.0” and highlight it (or use the search feature to search for “atom”). Click the install button and wait until the install has completed. Start the Atom text editor from the desktop.

On your own PC / Mac: Feel free to use a different editor, but if you want to use Atom, you can find it at :

https://atom.io

From Atom’s File menu, create a new file and in the editor window, enter the C source for the Hello World program (see the lecture notes for details). Save it into your COMP281 folder as helloworld.c (in the example below, the quotes have been made into “smart” ones by this text editor. You want to use normal quotes. If you copy and paste the text below, you’ll have to replace them with standard ones.)

```
#include <stdio.h>
int main(void)
{
        printf(“Hello, World!\n”);
        return 0;
}
```

Notice that once the file is saved with an extension, Atom figures out that it’s a C program and syntax highlights the code.

In the Cygwin window, type `ls` (lower case L and lowercase S - the unix list files command) and make sure that you can see the helloworld.c file listed there.
Now invoke the C compiler on the file using the command

```
gcc helloworld.c
```

If you’ve got no syntax errors in your code, it should compile, link and produce an a.exe program (Windows) or an a.out program (on Mac or linux). Note that on a real Linux system we’d get an a.out program, but since Cygwin is running on Windows, executable programs must have a .exe extension.

Execute your program by typing
./a.exe
It should run and produce the “Hello World!” message in the terminal window.
If that all worked as expected, try running the compiler with an additional flag as follows:

```
gcc -o helloworld   helloworld.c
```

You should see that there is a compiled executable program called helloworld.exe (on Windows, just “helloworld” on linux and Mac) in your COMP281 folder. Run it and see that it works as before. For future programs, make sure to use the -o option if you want to create executable programs with meaningful names.

OK, now I want you to try to program the solution to some simple problems. These problems are designed to be run by an automated system which provides input values and checks the output of your program. Your programs should not prompt for input values, or output additional text, beyond what is defined in the specification for the problem. Any unwanted text the you print means that the automated system cannot match your output with the output it expected to see.

On the COMP281 canvas site, open up the assignment for week 2 (called "Lab Work (week 2)"). Please note that the marks for this "assignment" do not count towards the overall mark for the module. This is just for you to try out CodeGrade.

You can use the Atom editor to create a program to solve this problem, and compile and test it as we did for the hello world program. Note that since CodeGrade doesn't want any other messages to be printed out, our program won't prompt us to enter the input values, it will simply sit there when running, waiting for us. If we were writing our program for a human to use we'd have additional printf statements to prompt the user (and code to check that the input values were within range). For this simple addition program we simply type three integer values on a line with a space between each, and then press return. The program should successfully add them all together and print the numeric result.

Once your program works as expected, you can return to your CodeGrade session in the web browser, and navigate to the window where you can submit your program code solution. Note: Your source code must be named **exactly** as described in the problem specifications, otherwise it will not be compiled and run.

For this practical, I'd like you to work your way through the following problems:

Week 2 part 1 - The Sum of A+B+C

Write a C program called **week2_1.c** to accept three integers, A, B and C (0 <= a <= 10, 0 <= b <= 10, 0 <= c <= 5) as inputs, and to compute and then print out their sum. e.g.

input: `3 4 5`

output: `12`

Week 2 part 2 - Record Marks

Write a C program called **week2_2.c** to record integer marks between 0 and 100 for a group of students. Input a list of marks. Input ends with 0 (0 itself is not someone's mark). Output the number of students who scored 1) greater than or equal to 85; 2) between 60 and 84; 3) strictly less than 60. e.g.

input: `88 71 68 70 59 81 91 42 66 77 83 0` output:

```
>=85:2
60-84:7
<60:2
```

hint: you'll need to use a loop for this.

Week 2 part 3 - Celsius-Fahrenheit Conversion Table

Write a C program called **week2_3.c** to print a Celsius to Fahrenheit conversion table. The program should accept three input integers and output an initial Celsius value, a step size, and the final Celsius value.

Output should be in the form of a table. e.g.

input: `-50 25 100`

output:

```
C=-50->F=-58
C=-25->F=-13
C=0->F=32
C=25->F=77
C=50->F=122
C=75->F=167
C=100->F=212
```

hint: you'll need to use a loop for this.