SCC110: Software Development Term 1. C Programming.

Problem Set 2: Getting Started with C

The Goal

This week's lab aims to:

- The goal for today is to get you really familiar with starting simple C projects and the edit→compile→execute cycle.
- There are three problems of increasing difficulty. There is also a more advanced 'hacker edition' if you are a more experienced programmer or complete all these exercises in good time.
- As a side effect, you will also be getting familiar with the lab environment, applications, and Shell – do make sure you've done the UNIX tutorial **FIRST**

Problem 1: "I love SCC.110"

Your task is to create a derivative of 'hello world' from Monday's lecture – and get it compiling, fixing any errors on the way.

The Task

To get you started, we'll work on a problem adapted from the `hello world' program that we showed you in Monday's lecture.

1. Start a new program (create a new text file on UNIX, e.g code helloworld.c, or by launching code and saving as helloworld.c)
2. Type not copy & paste the `hello world' program code. We recommend you type this as typing helps you remember the program structure and learn the correct syntax of the C language. You will make mistakes this way and it's slower, but learning to fix them is as if not more important than not making them in the first place!
3. Save the program
4. Ensure that your program compiles (fix any errors flagged by the compiler using the editor, and compile again until no errors are generated). To compile use:

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

5. Now adapt the program to print `I love SCC110'.

Problem 2: "Odd Numbers"

Start a new program (create a new file and choose a meaningful filename for your program).

The aim of the program is to print all the odd numbers between 1 and 100. e.g.:

Remember that the filename of C source code always needs to end in `\texttt{.c}'. *The editor will also recognise you're editing C and highlight the syntax of what you type, which is nice!*

The aim of the program is to print all the odd numbers between 1 and 100. e.g.:

```
1
3
5
7
```

You will need a '`main`' function. We recommend you consult the lecture slides, and particularly the slides on `while` and `for` loops. Naturally, there's plenty more info in the book which is a free download from the library! *You should get used to looking up language features for yourself!*

Problem 3: "Drawing a Square"
Our third and final problem for today is to print out a square using text characters. You should set a side length in your program and print out an appropriately sized square. A side length of 3 would output:

```
***
***
***
```

whereas a side length of 5 would yield:

```
*****
*****
*****
*****
*****
```

For additional credit and challenge, create a version of your program that prints a `hollow' square, thus:

```
***
* *
***
```

Hacker edition
If you've completed the other three tasks and want to push further, try this:

- Instead of printing a square, create a new program that prints a triangle of a height entered by the user.

Your program should 'pad' the output using spaces, i.e. a triangle of height 5 should look like:

```
    *
   ***
  *****
 *******
*********
```

How to Compile your code
You will need to include a standard header file for the *stdio* library (this defines what the `printf` function, among others, looks like – and without #include-ing this as we did in the lecture, you will get a compiler warning. You should do this by placing the following lines at the start of your program.

```
#include <stdio.h>
```

You then compile with something like:

```
      gcc -o outputfile sourcefile.c
```

This compiles a source file called `sourcefile.c` to an executable file named `outputfile` that you can execute by typing `./outputfile`. The './' is the 'path' so the shell can find outputfile (it's shorthand for look in the current working directory, more on this another day!).

If you don't specify an output file name with '-o' then you get 'a.out' as the name of your executable by default.

References

For a description of the stdio library see either http://en.wikipedia.org/wiki/C_file_input/output or the textbook, though note that there is way more detail than you need listed on this page!