Exercise 2: using make

You are going to use *make* to automate the building of your hello world program.

Create your Makefile

Using your favourite editor, create a file called `Makefile`
(note the capital 'M', which is normal practice here)

```
hello: hello.c
        gcc -Wall -o hello hello.c
```

There is one critical aspect here: the space at the front of the second line must be a single <TAB>. Do not use normal spaces. This is the most common error made with makefiles.

The rule you have written says:

* This is a rule to build *hello* (the target)
* This target depends on *hello.c*; that is, if hello.c changes, then hello needs to be rebuilt
* It gives the command needed to rebuild hello from hello.c

Now you can use it to rebuild your program - but it won't be rebuilt unless it is necessary to do so. You can force it to be rebuilt using 'touch': this resets the last-modified time on a file, so that it looks like you've edited it.

```
$ make
'hello' is up to date
$ touch hello.c
$ make
gcc -Wall -o hello hello.c
```

Make a change and rebuild

Edit hello.c and make a change to your program. For example, you can change the string which it prints from *Hello, world!* to something else.

Now rebuild it using make:

```
$ make
gcc -Wall -o hello hello.c
$ ./hello
Your new message
```

Notice how using 'make' makes life easier for the programmer, by issuing the correct command to recompile the program.