CS2106: Lab 3 (Not graded) - Understanding `main(...)`

This lab is **not graded**, as it should be quite straightforward. Nevertheless you **should** try it out. The correctness of your code should be self-explanatory from the output. There is no deadline as it is not graded but you should complete it before Lab 4. Do not submit any files.

Exercise 1: Learning about arguments to a process

Goals: We want to be able to understand arguments passed to an executable in Unix

The purpose of passing arguments to the program being executed is usually to give some additional information such as command line options, filename, etc. We already saw with `gcc` the use of options such as "`-c`", "`-o`", etc.

Write a program, `lab3-ex1.c`, to display the arguments passed to a process when it is "exec-ed" (with `execve()` or related calls).

As usual, the assumption with most labs is that the executable is compiled with the same name as the source code, e.g. `lab3-ex1`. Experiment with different arguments passed to `lab3-ex1` when you run the following from a command line shell:

- (a): `$ ./lab3-ex1`
- (b): `$ ./lab3-ex1 1 2`
- (c): `$ ./lab3-ex1 1 "2 3" 4`
  Note the quotes above. How many elements are there in array `argv[]`?
- (d): `$ ./lab3-ex1 "user = $USER"`
  The `$USER` illustrates the syntax used by the command line shell to replace an environment variable by its value (the environment variable `USER` can be seen in Exercise 3).

From this exercise, you should understand how `argv[]` and `argc` are set up.

Exercise 2: Learning about environment variables in Unix

Goals: We want to be able to understand environment variables which can be accessed by a process

Read the man page for the `printenv` command. Write your own version of `printenv`, `lab3-ex2.c`, which simply prints all the environment variables. In addition, your program can take one (or more) argument which specify which environment variable names to print.

Compare the results of:

- "`printenv`" with "`lab3-ex2`"
  Do they have the same content? Have a look at some of the environment variables, e.g. what is `LANG`, `PWD`, `PATH`?
- "`printenv USER`" with "`lab3-ex2 USER`"
  If your code is working, you should get the same result.
- "`USER="Mickey Mouse" printenv USER`" with your own `lab3-ex2`, e.g.:
  `$ USER="Mickey Mouse" ./lab3-ex2 USER`
  Do you understand what is going on? Run `printenv` again to see what is the value of `USER` now.

If everything is working, `congratulations`! You have written your own Unix utility, a clone of `printenv`. We see that it may not be difficult to write your own utility programs. Some programs may have more fancy features but the basic part may not be difficult to write.