SCC110: Software Development Term 1. C Programming.

Problem Set 3: Games Time!

The Goal
This week's lab aims to:

- Get you really familiar with adjusting program control flow using loops and conditions
- It also exercises your problem decomposition and solving abilities, specifically managing the 'game state' of two simple games
- There are two problems of increasing difficulty. There is also a more advanced 'hacker edition' if you fancy more of a challenge!

Problem 1: "Guess the Number"
This week we are building two versions of a simple but popular game. In the first, the computer "*thinks of a number between 0 and 9*" and then prompts the user to guess what it is.

The Task
Gameplay is simple. If the user guesses too high, the computer tells them to guess "*lower*"; if the user guesses too low the computer tells them to guess "*higher*". The game ends when the user guesses the number correctly. An example of the game play is shown below (where the user enters their number at the '> ' prompt for input):

```
I've thought of a number.
Enter your guess.
> 5
Higher
> 7
Lower
> 6
Well done, the number I thought of was 6.
```

1. Start a new program (create a new text file on UNIX, e.g code guess.c, or by launching code and saving as guess.c)
2. Think carefully about how to structure you code using loops and selection statements. You naturally want to pick a random number using rand() otherwise the game won't be very challenging!
3. A nice addition is to count how many guesses it took the player to guess the number and print this at the end.

Problem 2: "Higher or Lower"
The second game is based on a similar idea to the first, but is a little more complex. For this game the idea is that the computer displays a number between 0 and 9 and then asks the user to guess if the next number the computer picks will *be higher or lower*. If the user guesses correctly (or the numbers are the same) then the game continues to the next round, if the user guess incorrectly the game ends and the player loses. An example of the game play is shown below:

```
Higher or lower than a 7 (Enter 0 for higher and 1 for lower).
> 1
Higher or lower than a 4 (Enter 0 for higher and 1 for lower).
> 0
Higher or lower than a 5 (Enter 0 for higher and 1 for lower).
> 1
Game over, the number was 9.
```

We've deliberately used 1 for lower and 0 for higher so that you can use the input method we showed you in the lecture. Input in C can be a challenge, more on this in later weeks.

We recommend you consult the lecture slides, and particularly the slides on `while` and `for` loops, and where we talked about scanf(). 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!*

Hacker edition
If you've completed the both games and want to push further, try this:

- try creating a two player version of `guess the number'
- The human player should alternately take turns with a computer player in guessing the number
- You will need to handle the logic where the human player loses to the computer. A nice touch would be to `toss a coin' to see whether the human or the computer starts first
- Ideally the computer should learn from the player's guesses, just as the player learns from seeing the computer's attempts.

Approach this week
Think about the overall design of your program. The second game in particular needs some care, so we *highly recommend sketching out the algorithm on paper first*.

The following lines show you how to generate a random number in C:

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

int main ()
{
   int card;

   /* Do this only once at the start of your program,
      otherwise your sequence of numbers (game) will
      always be the same! */

   srand(time(NULL));

   /* Choose a number between 1 and 10 - we've called this 'card'
      because the number represents a card from ace to a ten in this
      program (rand() produces a large random int, we find the
      remainder from diving by 10 using '%' mod operator, then add 1
      so the card can't be 0) */

   card = rand() % 10 + 1;
   printf ("It's a %d.\n", card);
}
```

References
You can use the command man in the terminal to call up the `manual page' on many C functions printf. You can also search the manual using ``apropos <topic>''. *To get key help in the manual*

press `h', to quit press `q'. All the UNIX manual pages are online, so google man printf is also effective.