Week 10 - Introductory Programming Lab

Exercise 8: `for` Loops

Background

A loop that counts up (or down) is the most common type of loop, so most programming languages have a short-cut: the `for` loop.

This loop squishes the initial setting, conditional "do we continue", and ending "how to change the variable" into a single line. Given a statement of the form:

```
for (init; run_loop?; after_loop) {
    ...
}
```

- *init*: This is executed **once**, at the beginning of the entire loop.
  Generally will be something like `i=0`.
- *run_loop?*: The computer tests this **before each** loop.
  Generally will be something like `i<10`.
- *after_loop*: the computer executes this statement **after each** loop.
  Generally will be something like `i++`.

Technical details

The most common form of loop is this:

```
#include <stdio.h>

int main()
{
     int i;
     for (i=0; i<10; i++) {
         printf("The loop has repeated ");
         printf("%i times.\n", i);
     }
     getchar();
     return 0;
}
```

Here's a weird loop:

```
#include <stdio.h>
int main()
{
     int i;
     for (i=2000; i>0; i=i/2-3) {
         printf("What is this loop ");
         printf("doing?  i is %i\n",i);
```

```
    }
    getchar();
    return 0;
}
```

Your task...

Back in the good old days, there were no fancy graphics in computer games; we used text to represent everything. Your task is to draw a room from a family of games called *Roguelikes* -- the player (represented by the `@` symbol) must explore a dungeon.

2x2 room, player at 0,1:

```
+--+
|..|
|@.|
+--+
```

5x3 room, player at 1,2:

```
+-----+
|.....|
|.....|
|.@...|
+-----+
```

14x8 room, player at 8,5:

```
+--------------+
|..............|
|..............|
|..............|
|..............|
|..............|
|........@.....|
|..............|
|..............|
+--------------+
```

Your task is to write a function (with extra "helper" functions) to draw such a room.

- How big are the rooms? What is the coordinate system?
- Your `int main()` must contain only:

```
int main() {
     drawRoom(2,2,0,1);
     drawRoom(5,3,1,2);
     drawRoom(14,8,8,5);

     getchar();
     return 0;
 }
```

- In addition to `drawRoom()`, write *three extra* functions. The first draw a horizontal `+----+` line, the second draws a line without the player `|....|`, and the third draws the line with the player `|..@.|`. The `drawRoom` function should call those other helper functions when appropriate.
  (examples of the three extra functions are not drawn to scale)
- Use `for` loops. You may *not* use `if` or `while` in this exercise.

(optional: combine this exercise with keyboard input -- let the player move around in the room, bump into walls, etc. Ask the user to turn on the *numlock* key and to press `enter` after every move, then you can read his moves by reading `int` from the keyboard. Use a while loop for this movement.)

Show your work to a demonstrator/GTA.

Exercise 9: Arrays

Background

An *array* is an ordered sequence of values; you cannot rearrange values without changing the meaning. A two-dimensional array is just a normal "table".

In C, arrays are indexed (accessed) starting from 0. For example,

```
int array[4] = {3, -2, 987, 12};
```

creates an array in memory, storing each value at the appropriate index:

Index	0	1	2	3
Value	3	-2	987	12

The array's size can only be specified when you create it. If you want to store more information in that array, you need to create a new (bigger) array, then copy information from the old array to the new array.

Due to the way C handles arrays and pointers (coming later), you can modify an array inside a function without needing to return anything. The function needs to know the array's size, though!

Technical details

Initializing and displaying:

```
#include <stdio.h>

int main() {
    int array[8] = {1, 2, 8, 3, -5, -1, 1};

    int i;
    // display the array
    for (i=0; i<8; i++) {
        printf("%i ", array[i]);
    }
    printf("\n");

    // change the array
    for (i=0; i<8; i++) {
        array[i] = 2 * array[i] - 1;
    }

    // display the array again
    for (i=0; i<8; i++) {
        printf("%i ", array[i]);
    }

    printf("\n");
    getchar();
   return 0;
}
```

Arrays and functions:

```
#include <stdio.h>

void printArray(int arr[], int size) {
    int i;
    for (i=0; i<size; i++) {
        printf("%i ", arr[i]);
    }
    printf("\n");
}

void changeArray(int arr[], int size) {
    int i;
    for (i=0; i<size; i++) {
        arr[i] = arr[i] * arr[i] + 3;
    }
}

int main() {
```

```
    int array[8] = {1, 2, 8, 3, -5, -1, 1};
    printArray(array, 8);
    changeArray(array, 8);
    printArray(array, 8);

    getchar();
    return 0;
}
```

Your task...

The Fibonacci numbers are a famous sequence of numbers. They begin with 0 and 1, and then the next value in the sequence is the sum of the previous two values.

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765

(fib[8] is 21 -- remember to start counting from 0!)

Write a program that calculates the Fibonacci sequence.

- Make a function that accepts n, which is the number of integers to generate.
- Declare an array, initialize it with only the first two Fibonacci numbers, then calculate the rest.
- Display the sequence for n=10 and n=20.
- Try generating output for n=50. If anything goes wrong in this step, you don't need to fix it. Just add a comment explaining what happens, and why.

Show your work to a demonstrator/GTA.