Conditions & Loops

Exercise 6: if conditional

Background

The logic of an `if` statement is fairly simple. If the statement is true, the *code block* which is indicated by a statement or the contents of a `{...}` block) is run. A closely related command is the `if ... else` statement, which acts precisely as we would expect.

Here are two cautions to avoid common problems.

1. In C, `x = 1` and `x == 1` perform completely different tasks:
    - The single `=` is an *assignment* operator: it changes to the value of x to be 1.
    - The double `==` is the *comparison* operator: it returns true if x is equal to 1.
2. If we forget to use the `{ ... }` code block, then it will be very easy to make a mistake later on. The computer will happily execute statements which we did not think would be running. This is illustrated at the bottom of the "conditional" example.

Technical details

`Conditional:`

```
#include <stdio.h>

int main()
{
    int x = 3;

    if (x == 4)
    {
        // this line will not be printed
        printf("x is equal to 4.\n");
    }
    // one of these two lines will be printed
    if ((x > 10) && (1 != 0))
    {
        printf("Expression is true.\n");
        printf("x is greater than 10.\n");
    }
    else
    {
        printf("Expression is false.\n");
        printf("x is not greater than 10.\n");
    }
```

```
    // something weird happens here!
    if (x > 10)
        printf("Expression is true.\n");
    printf("x is greater than 10.\n");

    return 0;
}
```

The logical operators (not `!`) (and `&&`) (or `||`) are particularly useful while using the `if` statement. Occasionally the exclusive or ^ is also useful. Parentheses `()` are highly encouraged when using logical operators.

Random numbers

Background

In engineering or computer science, when people write "random number" with regards to computers, they mean "pseudorandom number". The basic problem is that if we start from a single value (known as the *random seed*), and perform a series of mathematical operations on it, we will not get an actual random number -- every time we start from the same random seed, we will get the same number!

To quote one of the giants in computer science,

> *"Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin."*
> *John von Neumann*

However, pseudorandom numbers are good enough for this course.

Technical details

Generating a random number:

`// getRand()` requires extra `#include` files!

```
#include <stdio.h>
#include <stdlib.h>  // extra includes!
#include <time.h>

/* Get a random number from 0 to 0.9999999
    ***** DON'T MODIFY THIS FUNCTION *****
*/
float getRand() {
     return rand() / (RAND_MAX+1.0);
}

int main() {
     srand( time(NULL) ); // initialise the process
     getRand(); // kick-start the random numbers

     float number = getRand();
     printf("Random number: %f\n", number);
     return;
```

```
}
```

Example

Create a "guessing game" program. The program should:

- Make the computer pick a random number to be the answer. It should be an `int` between `1` and `32` (inclusive).
- Use the `getRand()` function. Instead, you should take the value it returns (a `float` between 0 and 0.999...) and do some math to transform that into an `int` between 1 and 32).
- Write a *function* for the user's guess. This function must:
  - have one integer argument (`int correct_answer`),
  - read an `int` from the keyboard,
  - check the user's `int` against the correct answer,
  - output the appropriate message ("correct" / "too high" / "too low"),
  - returns a 1 if the user's guess was correct, and `0` if the user's guess was wrong.
- Give the user 5 chances to guess the right answer.
- End the game if the user is correct, OR they have used up all 5 chances. Print either a "you win" or "you lose" prompt accordingly.

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

// Random number generation
float getRand()
{
    return rand() / (RAND_MAX + 1.0);
}

// guessing function where we pass the correct answer as an argument
int guess(int correct_answer)
{
    int count = 0;
    while (count < 5)
    {
        printf("guess the number from 1 - 32:\t");
        int i;
        scanf("%d", &i); // enters the value to be guessed
        if (i == correct_answer)
        {
```

```
            printf("correct\n");
            printf("you win\n");
            return 1;
        }
        else if (i < correct_answer)
        {
            printf("too low\n");
            if (count == 4)
            {
                printf("you lose\n");
                printf("0");
                return 0;
            }
            count++;
            continue;
        }

        else
        {
            printf("too high\n");
            if (count == 4)
            {
                printf("you lose\n");
                printf("0");
                return 0;
            }
            count++;
            continue;
        }
    }
    return 0;
}

int main()
{
    srand(time(NULL));                  // init random
    float a = getRand();                // kick-start the random
numbers
    int correct_answer = ceil(a * 32); //the ceiling function
ensures we get an integer
    guess(correct_answer);
}
```