A Mysterious Parody Bit

Lab 1 - COMP211 - Fall 2020

What truth lies in the center of a zero?

```
        ======================BEGIN TRANSMISSION====================
22537472616e67652063617365206 6726f6d2074686520737461727420412063617365207 76974
68206120686F6C6520696E20746865206D6964646C652E20200A204120646F7567686E75742E2220
7e2042656e6f697420426c616e630a0a5061727420313a20687474703a2f2f6269742e6c792f6635
3636383964313966326165363864386234333361363962616432663333343537666137303537 0a
        ====================== END TRANSMISSION ====================
```

In this lab you will follow a crumby trail of nibbles and clues, starting with the encoded transmission above. Bit-by-bit you will shift your way closer to the elusive truth. In the end, you will emerge a bitwiser.

Accept the following GitHub classroom assignment: https://classroom.github.com/a/6wpWvRnp

With your project repository open, click the green "Clone or Download" button. Copy the HTTPS URL. Back in a `learncli` container, clone this url with the `git clone <URL>` subcommand. Replace `<URL>` with your copied URL. (Right click to paste in Windows.) After cloning, change your working directory to the cloned repository.

Part 0 of 3. A hex spell reversed is `xeh`

The encoded transmission is found at the path `clues/00-nibbles.hex` in your repository.

The original, unencoded message was a sequence of ASCII `char` bytes. This data was then encoded to hexadecimal digits encoded as ASCII `char` bytes. Your job is to write a program to read in the encoded data and output the original, unencoded message.

Before you begin programming, you should be able to answer the following questions confidently:

1. What is the difference between a `char` value your program receives as input, which just so happens to be in the ASCII ranges of `0-9` or `a-f`, and the numerical value of the hexadecimal digit that same ASCII `char` *represents*?
2. How many hexadecimal digits do you need to decode a single ASCII `char`?
3. What programming calculations or bitwise operations do you need to perform in order to combine hexadecimal values together? You should start with arithmetic based on the formula presented in class. For full style credit, though, you should rewrite this arithmetic using bitwise operators only.
4. What is the bitfield difference between uppercase and lowercase letters?

Name your C source code file `xeh.c`. Its purpose is to decode the hex message above, or any other message encoded like it. You can add this file to your repository's root directory. **You *must* perform the representation conversions in your own source code using bitwise operators, and may not rely on any library functions.** You can assume the input will contain only an even number of ASCII-encoded hex digits. Notice the hex letters can be either upper or lower case in the encoding. The only other input character you must handle is the new line character, which your program should ignore. Be sure `xeh.c` has the course header lines:

```
// PID: 123456789
// I pledge the COMP211 honor code.
```

To compile your program to an executable named `xeh`, instead of `a.out`, so you can run it with `./xeh`, use the following `gcc` options (the `-o xeh` option is the one which specifies the output binary name):

```
$ gcc -Wall -Wextra -std=c11 -g -o xeh xeh.c
$ cat clues/00-nibbles.hex | ./xeh # Pipes the encoded message to your program.
```

Once you've successfully decoded `clues/00-nibbles.hex`, you will know have the clue to bring you to Part 1.

Grading

You must include the course header in all `.c` files. Style is manually graded by course staff, after the late deadline, on the following criteria:

- Use of curly braces around *all* control statement bodies (`if`, `while`, `for`, etc)
- Proper indentation (use the following `vim` trick: `1G=G`)
- All magic numbers are defined as constants
- No global variables
- For parts 0 and 1, only bitwise operations were used in the key conversions between hex and ASCII
- No libraries outside of `stdlib.h` and `stdio.h` used.