Mysterious Parody Bits

Lab 1 - COMP211 - Fall 2020

> “Education never ends, Watson. It is a series of lessons, with the greatest for the last.”
>
> ~ Sherlock Holmes, *The Red Circle*

Part 1 of 3. A `xeh` spell reversed is `hex`

In this part of the lab you will write `hex.c` to convert plaintext ASCII to its hexadecimal representation.

Create a file named `hex.c` in the root directory of your project’s repository. Its purpose is to read input as plaintext ASCII, until `EOF` is reached, and produce each character’s hexadecimal representation in ASCII characters `0-9` and `A-F`. Here is an example usage:

```
$ echo "Elementary, Watson!" | ./hex
456C656D656E746172792C20576174736F6E210A
```

`hex.c` is the complement of `xeh.c`. Passing one’s output into the other’s input, via a pipe, should give you the first’s input:

```
$ echo "Elementary, Watson!" | ./hex | ./xeh
Elementary, Watson!
$ echo "456C656D656E746172792C20576174736F6E210A" | ./xeh | ./hex
456C656D656E746172792C20576174736F6E210A
```

Your generated hex output should end with a newline character. If the hex output is 80 characters or longer, then it should “wrap” the output by inserting new line characters after every 80 hex digits.

```
$ echo "The world is full of obvious things which nobody by any chance ever observes." | ./hex
54686520776F726C642069732066756C6C206F66206F6276696F7573207468696E67732077686963
68206E6F626F647920627920616E79206368616E63652065766572206F627365727665732E0A
```

As with `xeh`, you should utilize `gcc`’s ability to generate executables named something other than `a.out` with the `-o` option:

```
$ gcc -Wall -Wextra -g -std=c11 -o hex hex.c
```

Finally, for full style credit you should use bitwise operators instead of arithmetic. Starting with arithmetic is encouraged, however.

The game is afoot!

Once your `hex` program is working, hex encode the following string: `the game is afoot`

Then, use the generated hex string to navigate to the following URL, substituting the placeholder with your encoded string.

`http://bit.ly/your-encoding-here`