Assignment 1 — Binary and Hexadecimal

Due: Monday, January 10th

Due to their ease of electronic implementation, modern computers represent values in binary, which, for human readability, are often written in hexadecimal. When working with these values, computers do not have the luxury of converting them to other types.

Deliverables:

Required Files: `binary.py`, `hexadecimal.py`

Optional Files: *none*

Part 1: Ground Rules

Because binary and hexadecimal values are so prevalent in computer science, languages such as Python already include built-in functions and operators for working with them. Throughout this assignment, you may *not* use any of the following:

- Type Conversion (including, but not limited to, `bin`, `hex`, `int`, `float`, `bool`, and `str`)
- Bitwise Operators (`~`, `&`, `^`, `|`, `<<`, and `>>`)
- Binary Sequences (`bytes`, `bytearray`, and `binascii`)

…rather, binary "numbers" will be represented as Python strings, and you should implement all functions by comparing their individual characters to `"0"` and `"1"`.

Part 2: GitHub Classroom

In this class, you'll submit your programs through GitHub, an online host for tool known simply as "git". Git tracks changes you make to files so that you can easily undo or combine changes while working on a project.

You should already be familiar with the basic features of git from previous classes, such as CSC 202 and possibly CSC 203. If not, the following resources might be helpful:

- `https://help.github.com/en/articles/set-up-git`
- `https://help.github.com/en/articles/cloning-a-repository`
- `https://help.github.com/en/articles/managing-files-using-the-command-line`
- `https://help.github.com/en/articles/pushing-to-a-remote`

GitHub Classroom allows your instructors to accept electronic submissions through GitHub.

1. Enter your Cal Poly and GitHub usernames in this form: `https://goo.gl/jTK3Bx`. This allows us to determine who should get credit for your electronically submitted assignments.
2. Accept the GitHub Classroom assignment by clicking on the GitHub link in Canvas. This will create a new repository for you on GitHub which you will use to submit this assignment, as well as provide some starter code and sample test cases.

Part 3: Binary Operations

Complete the functions in the given `binary.py` file. The first three functions concern addition, negation, and subtraction of binary numbers. Do *not* convert the binary numbers to decimal (or to any other base) in order to implement these three operations. After all, a real computer would not do that; that would defeat the purpose of using binary in the first place.

Rather, computers check each individual bit of the operands, and thereby determine the individual bits of the result. For example, when adding two binary numbers:

```
  00100110                    38
+ 00000111   equivalent to  +  7
----------                  ----
  00101101                    45
```

…we can begin by applying three basic rules:

- If neither addend contains a '`1`' in bit n, the sum contains a '`0`' in bit n.
- If exactly one addend contains a '`1`' in bit n, the sum contains a '`1`' in bit n.
- If both addends contain a '`1`' in bit n, the sum contains a '`0`' in bit n, and a '`1`' is carried to bit $(n+1)$.

These rules alone[1] will not suffice, because there is the additional possibility of having carried a bit from the previous column, bit $(n-1)$. Consider how these rules can be extended to account for such cases[2].

Once the `binary.add` function is completed, the `binary.negate` function should be relatively straightforward. The `binary.subtract` function should then be trivial.

Part 4: Hexadecimal Operations

Complete the functions in the given `hexadecimal.py` file.

Part 5: Testing

A minimal set of unit tests has been provided for both your binary and your hexadecimal functions. They can be invoked from the command prompt, for example:

```
>$ python3 binary_tests.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.000s

OK
```

These tests are by no means exhaustive, and you are encouraged — though not required — to write additional unit tests. The quality of your tests will not be assessed as part of grading.

Part 6: Submission

The following files are required and must be pushed to your GitHub Classroom repository by the deadline:

- `binary.py` — A working implementation of functions for binary numbers, as specified.
- `hexadecimal.py` — A working implementation of functions for hexadecimal numbers, as specified.

The following files are optional:

- *none*

Any files other than these will be ignored.

[1] This is called a *half adder*: `https://en.wikipedia.org/wiki/Adder_(electronics)`
[2] The hardware circuit which you are essentially replicating in software is called a *full adder*.