Assignment P4

Due Date: November 5

Purpose

It's time to think like a computer scientist! That means, you are now a developer *for* Python, not just a *user* of Python. As a developer, you are tasked with implementing versions of the built-in `int()` and `bin()` functions, which can be used to convert binary values to integers and vice-versa, respectively. As a developer, you have to implement these without using the built-in functions. To complete the project, you will also use repetition statements, additional selection statements, and you will get data from a file rather than from the keyboard.

Problem

Every computer must translate the user's input, whether it be directly from the keyboard or through some software, to binary. Conversely, the computer translates its binary data back to something the user can understand. In this program, you will convert integer values in base 10 to binary (base 2), and from binary to decimal. We have seen algorithms on how to do this on paper, but now those methods must be translated into a true Python program.

Input

The program should first prompt for the name of a file (one word, no spaces).

The file will contain any number of lines. Starting from the first line, every **other** line will contain an integer between 1 and 3, inclusive. These numbers represent the following functionality:

1	Convert binary number to decimal
2	Convert decimal number to binary
3	Quit

If the input is a 1, the next line in the file will then be a binary value. If the input is a 2, the next line will contain a decimal (integer) number. In either case, you may assume that the file contains valid values; that is, the data will not have extra spaces, unwanted characters, etc. Note that negative values are acceptable.

Once a conversion is completed, the whole process should repeat, until such time as the program reads a 3 from the file, which indicates there is no more data. You may assume that there will be no incorrect values in the input file.

For example, the contents of a valid file would be:

```
2
534
1
1101
3
```

Output

The program should display the original values and their corresponding conversions. When the program finishes, there should be a message to that effect. For example, for the input file above, the output would be similar to:

```
The binary equivalent of 534 is 1000010110
The decimal equivalent of 1101 is 13
All done!
```

Specifics

- You must read a file in the **exact** form specified above! We will run tests using this format, and if your program does not read a file properly, your grade will suffer.
- You must use at least one `for` loop and one `while` loop.
- You must write a function for prompting the user and returning the file pointer (variable). The function should not have any parameters.
- You must write a function called `int(`x`)` which will return the integer value of the binary parameter x.
- You must write a function called `bin(`x`)` which will return the binary value of the integer parameter x.
- At the beginning of each function, just below the header, write a comment describing its purpose, its parameters, and what it returns.
- Do not use any global variables.
- All input represents integer data only; that is, no data will have a decimal point.
- You may assume that all values in the file will be correct.
- All output should be clearly labeled.

Notes

Do not go running to the computer! You must first think about how to solve the problem. Be sure you can work the problem out on paper before you try to write a program! This project is not particularly long, but it may test your problem solving skills.

You may find it easier to work with strings as input rather than integers.[1]

I suggest using `readline()` to read each line of the file. Be aware, however, that when you are reading a string, the string's last item will be the (invisible) newline character.

Do not try to write the entire program at once! Write one portion at a time. For example, you might first get your input to work properly. Reading the data properly is always the first step in any program! There is no sense in continuing if you are not sure that you have correctly read the data. Then you might add in the portion of the code that will do **one** of the conversions. When that works, add in more code to find the other conversion.

As usual, send your completed program to me via email. Hand in a printout of your source code in class on November 6^{th}. Don't forget to write and **sign** the Wheaton Honor Code pledge.

In mathematics you don't understand things.
You just get used to them.
– Johann von Neumann

[1] Hmmm, this seems to be an interesting "note"...