Lab 4 — Decimal/Binary Conversion

1. Using our **CharFileReader** and **CharFileWriter** classes, finish the implementation of the **CopyTextfile.java** program. This program should accept two command-line arguments: the name of a source text file and the name of a destination text file. It should then copy all of the characters from the source to the destination. If the wrong number of command-line arguments is given, a warning message should be printed. Test your program as shown below:

```
> java CopyTextfile haunting.txt
Usage: java CopyTextfile <source> <destination>

> java CopyTextfile haunting.txt haunting2.txt
Read 504 chars from haunting.txt
Wrote 504 chars to haunting2.txt
> java diff haunting.txt haunting2.txt
Files are identical

> java CopyTextfile empty.txt empty2.txt
Read 0 chars from empty.txt
Wrote 0 chars to empty2.txt
> java diff empty.txt empty2.txt
Files are identical
```

2. Write a program called **Decimal2Binary** that accepts a single command-line argument, which should be an integer in the range 0-255, and converts it to an 8-bit binary value. Your program should construct an `int[]` array of eight 0s or 1s representing the bits and then print out the array contents in the format shown below, using `Arrays.toString`. If the user provides a wrong number of arguments, or a value not in the range 0-255, the program should print an appropriate warning message. Below are some examples of what the output of your program should look like:

```
> java Decimal2Binary 42
42 in binary is [0, 0, 1, 0, 1, 0, 1, 0]

> java Decimal2Binary 115
115 in binary is [0, 1, 1, 1, 0, 0, 1, 1]

> java Decimal2Binary 255
255 in binary is [1, 1, 1, 1, 1, 1, 1, 1]

> java Decimal2Binary 256
Sorry, value must be in the range 0-255

> java Decimal2Binary -5
Sorry, value must be in the range 0-255

> java Decimal2Binary
Usage: java Decimal2Binary <0-255>

> java Decimal2Binary 10 20 30
Usage: java Decimal2Binary <0-255>
```

(continued on next page)

3. Write a program called **Binary2Decimal** that accepts a binary number as a single command-line argument, written as a string of bits (0s and 1s), and converts it to an unsigned decimal integer. The string may be *of any length*, and may include leading 0s. You are not allowed to use `Integer.parseInt`, `Integer.valueOf`, or `Math.pow` for this exercise (for reasons that will become clear later). If the wrong number of arguments is given, the program should print an appropriate warning message. Below are some examples of what the output of your program should look like:

```
> java Binary2Decimal 101010
101010 in decimal is 42

> java Binary2Decimal 0000101010
0000101010 in decimal is 42

> java Binary2Decimal 1111
1111 in decimal is 15

> java Binary2Decimal 11100011
11100011 in decimal is 227

> java Binary2Decimal 1011011101111
1011011101111 in decimal is 5871

> java Binary2Decimal
Usage: java Binary2Decimal <bit_string>

> java Binary2Decimal 1 0 1 0 1
Usage: java Binary2Decimal <bit_string>
```