**Bitwise**

Bitwise operators work on the level of bits, the binary digits that comprise "regular" numbers. Don't even think about proceeding before reading through the presentation in the "Info" folder.

1. Use bitwise AND (&) to determine if an integer is even. The AND operator will be used in lieu of modulus as in the "normal" algorithm you're familiar with.
   1. **Note:** the == operator has higher precedence than &; without round brackets, the integer comparison will happen prior to the & being applied; the & operator does not work for an integer and boolean (even though logically it could as, conceptually, a boolean is a single bit).
      1. [Here](https://introcs.cs.princeton.edu/java/11precedence/) is a link to an operator precedence cheat sheet that may be useful to bookmark.
2. To determine if a number N is a power of 2, repeatedly divide N by 2 if N is even. N is power of 2 if you end up with 1 (note that N = 0 is a special case; 0 is not a power of 2). Code this algorithm using no arithmetic operators.
3. Without looking back at the powerpoint, write the code to check if the ith bit of an integer is set (is a 1), where the index i starts at 0 from the right-most bit.

**Hint:** Any number 2N will have a 1 bit at index N (e.g. for N = 3, 23 is {1000}2,make sure you understand why this works) and can be accomplished with 1 << N.

1. Set the ith bit of a number (to 1).
2. Flip (toggle) the ith bit of a number.
3. Find the largest power of 2 less than or equal to a given number.
4. Copy/paste the numbers array in the **"Data\numbers.txt"** file into your program.

For each integer in this array, go through the number bit-by-bit, from left to right. For each bit, if the bit is a 0, print a blank space; if it is a 1, print the letter 'X'. You should print 32 characters on each line, with as many lines of output as there are numbers in the array.

1. Write a method static String intToString(int number) that will output all 32 bits of the given integer, separating groups of 4 bits with a space.

42 = 0000 0000 0000 0000 0000 0000 0010 1010

-1001 = 1111 1111 1111 1111 1111 1100 0001 0111

1. Once again using the numbers array, write a program to output the 32-bit integers in the array as 8-digit hexadecimal numbers, including the "0x" prefix that is used for hex numbers in Java. To get one of the hex digits in a number, grab a group of four bits from the number and convert it into the equivalent hexadecimal symbol. For example, the number 15 would be output as 0x0000000F.
   1. Java has built-in methods for doing this type of transformation, but you shouldn't use them for this exercise (though they may be helpful for checking your work).
2. Write a program to read binary numbers entered by the user and convert them to normal integers.
   1. Your program should read a series of lines of input from the user, as Strings. Use an empty line of input as a signal to end the loop.
   2. Aside from the empty line at the end, each line of input should contain between 1 and 32 zeros and ones. For example, if the user's input is "1101", then your output would be the number 13.

**Tip:** If you get stuck, first complete this exercise using normal arithmetic operators (multiplication, addition etc.), then convert your solution to bitwise operators.

1. Repeat the previous exercise, except now the user will enter hexadecimal numbers instead of binary.
   1. Each line of input from the user should contain between 1 and 8 hexadecimal digits. (You can allow an optional "0x" at the beginning, if you want, but you don't need to require it.)