COP 3502 Recitation Sheet: Recursion

Directions: Each of these questions is from either a past Foundation Exam or one of my past exams. They were created to be written on paper. However, each of these can be coded on a computer and tested. I strongly recommend first coding on paper, but then transferring to the computer and testing the function until you are convinced it works.

Each of the following questions asks you to write a recursive function.

1) The code below returns the number of zeros at the end of n!

```
int zeros(int n) {
    int res = 0;
    while (n != 0) {
        res += n/5;
        n /= 5;
    }
    return res;
}
```

Rewrite this method ***recursively***:

```
int zeros(int n);
```

2) Write a *recursive* function that returns the sum of all of the even elements in an integer array *vals*, in between the indexes *low* and *high*, inclusive. For example, for the function call sumEven(vals, 3, 8) with the array vals shown below, the function should return 24 + 8 + 10 = 42, since these three numbers are the only even numbers stored in the array in between index 3 and index 8, inclusive.

Index	0	1	2	3	4	5	6	7	8	9
Vals[i]	15	13	28	19	24	8	7	99	10	14

```
int sumEven(int vals[], int low, int high);
```

3) Consider the following game. You are given a positive integer, *n*. Your goal is to change that integer to a number 0 or less. You are allowed two possible operations to reduce your number: (1) subtract 10 from it, (2) integer divide it by 3. Write a ***recursive*** function that calculates the minimum number of operations you have to apply in sequence to reduce your number to 0 or less. (For example, if your number was 19, you could divide it by 3 to obtain 6 and then subtract 10 to obtain -4, to obtain the goal in 2 steps. There is no way to do this in 1 step, so the correct answer is 2 for this case.) **Hint: recursively try both moves, see which one "wins" the game for you more quickly and build off that move.**

```
int minMovesToWin(int n);
```

4) Mathematically, given a function f, we recursively define $f^k(n)$ as follows: if $k = 1$, $f^1(n) = f(n)$. Otherwise, for $k > 1$, $f^k(n) = f(f^{k-1}(n))$. Assume that a function, f, which takes in a single integer and returns an integer already exists. Write a ***recursive*** function fcomp, which takes in both n and k ($k > 0$), and returns $f^k(n)$.

```
// Assume this is written already.
int f(int n);

// This is what you write.
int fcomp(int n, int k);
```

5) Write an ***efficient*** ***recursive*** function that takes in a ***sorted*** array numbers, two integers, low and high, representing indexes into the array, and another integer, value, and returns the index in the array where value is found in the array in between index low and high, inclusive. If value is NOT found in the array in between indexes low and high, inclusive, then the function should return -1.

```
int search(int numbers[], int low, int high, int value);
```