Name: _____
UCFID: _____
NID: _____

1) (10 pts) DSN (Recursive Coding)

Imagine a Towers of Hanoi puzzle with 4 towers, labeled A, B, C and D, with a tower of n disks, starting on tower A, to be moved to tower B, using the usual rules of the puzzle. One strategy to solve the puzzle would be to move the k smallest disks recursively to tower D, where all 4 towers are used. Then, with the remaining n - k disks, use the usual strategy (since tower D is unavailable), which will take exactly $2^{n-k} - 1$ moves, to transfer the bottom n - k disks to tower B. Finally, now that you can use all 4 towers again, recursively transfer the k smallest disks on tower D to tower B, completing the puzzle. Sonia has decided that she wants the value of k to be set at (3n)/4, using integer division. For this question, write a recursive function that takes in n, the number of disks in the game, and returns the number of moves that it will take to solve the game using Sonia's strategy. A function prototype with pre and post conditions is provided below. (**Note: In order to get full credit you MUST NOT USE the pow function in math.h because it returns a double which has inherent floating point error. Please manually use integers to calculate an exponent or bitwise operators.)**

```
// Pre-condition: 1 <= n <= 115 (ensures no overflow)
// Post-condition: Returns the number of moves Sonia's strategy
//                 will take to solve a Towers of Hanoi with n
//                 disks with 4 towers.
int fourTowersNumMoves(int n) {

    // Grading: 2 pts
    if (n == 1) return 1;

    // Grading: 2 pts to calculate this split somewhere.
    int split = (3*n)/4;

    // Grading: 1 pt return, 1 pt 2*, 1 pt rec call
    // 3 pts calculation of (2 to the power n-split)-1.
    return 2*fourTowersNumMoves(split) + (1<<(n-split)) - 1;


}
```