**Big Oh Analysis**

**4. Recursive Factorial**

Both of the implemented algorithms are O(n). Since they are recursive, we can use the recurrence relation where for , where c equates to the amount of constant work done per recursive call and as the base case. Since there are a constant amount of steps per step, we can translate the recurrence relation to O(n) time complexity.

**5. A portion of an array backward**

This function prints a portion of an array in reverse order by specifying an end index and printing each element back to a start index. This recursive formula subtracts 1 from a parameter until the base case is reached. The array portion will have recursive calls with constant work again, leaving us with the recurrence relation for where . Since this, too, has continuous work, is dependent on , and decrements by 1 every call, this is also O(n).