1. For each of the following recursive functions:

   • (1 point) Describe what the function computes (careful, some of these are tricky!)
   • (1 point) Give a recurrence relation that describes the running time of the function (Give both base and recursive cases)
   • (2 point) Solve the recurrence to get a Θ running time for the function. Use either the repeated substitution method, or the recursion tree method (which is essentially the same as the repeated substitution method, just a little more graphical). Do not use the master method for this question (you will have a chance to use the master method on later questions!)

   (a) int recursive1(int n)
       {
       if (n == 0)
       return 0;
       else
       return 1 + recursive1(n-1);
       }

   (b) int recursive2(int n)
       {
       if (n == 0)
       return 0;
       return recursive1(n) + recursive2(n-1);
       }

   Note that we are making both a recursive call and a non recursive call in this function! recursive1 is the function defined in question 1b.

   (c) int recursive3(int n)
       {
       if (n == 0)
       return 1;
       else
       return recursive3(n-1) + recursive3(n-1);
       }

   (d) int recursive4(int n)
       {
       if (n == 0)
       return 1;
       else
       return 2 * recursive4(n-1);
       }
(c) int recursive5(int n)
{
    if (n <= 1)
        return n;

    int dummy = 0;
    for (int i = 0; i < n; i++)
        dummy++;

    if (n % 2 != 0)
        return 1 + recursive5(n-1);
    return recursive5(n/2) + recursive5(n/2);
}

For this question, you should calculate what the function returns for all cases. For the runtime analysis, you may assume that n is a power of 2, so that you are always in the 2nd recursive case.

2. Use the substitution method (that is, proof by induction) to prove the following bounds:

(a) (4 points) $O(n^2)$ bound for:

\[
\begin{align*}
    T(0) &= C_1 \\
    T(1) &= C_1 \\
    T(n) &= T(n-2) + C_2 n
\end{align*}
\]