1. Give the $\Theta()$ running time of the following code fragments, in terms of $n$. Show your work! (Be careful, some of these are tricky!)

(a) for (i=0; i < n; i++)
    {
        for (j = n; j > 1; j--)
            sum++;
        for (j = n; j > 1; j = j - 3)
            sum++
    }

(b) for (i=1; i < n; i = i + 2)
    for (j = n; j > n / 2; j = j - 2)
        for (k = 1; k < n / 2; k = k * 2)
            sum++;

(c) for (i=1; i < n; i++)
    {
        for (j = 1; j < i; j++)
            sum++;
        for (j = 1; j < n; j++)
            sum++;
        for (j = 1; j < n; j = j * 2)
            sum++;
        for (j = 0; j < n; j = j + 2)
            sum++
    }
2. Consider the following function:

```c
int recursive(int n)
{
    if (n <= 1)
        return 1;
    else
        return recursive(n - 1) + recursive(n - 1) + recursive(n - 1);
}
```

(a) What does this function calculate?
(b) Give a recurrence relation \( T(n) = \ldots \) for this function (be sure to include both base and recursive cases!)
(c) Solve the recurrence relation to get the \( \Theta() \) running time of the function, in terms of \( n \). Show your work, using either repeated substitution, the master method, or a recursion tree.

```c
int recursive2(int n)
{
    if (n <= 1)
        return n;
    sum = 0;
    for (int i = 0; i < n; i++)
        sum++
    return recursive2(n/3) + recursive2(n/3) + recursive2(n/2) + sum;
}
```

(a) Give a recurrence relation \( T(n) = \ldots \) for this function (be sure to include both base and recursive cases!)
(b) Solve the recurrence relation to get the \( \Theta() \) running time of the function, in terms of \( n \). Show your work, using either repeated substitution, the master method, or a recursion tree.