function f(x,n)
{
    if (x > 0)
    {
        f(x/2,n);
        for (var i from 0 until n)
        {
            println(n);
        }
    }
}

So I'm thinking the space complexity is theta(1).

Because the recursive call occurs X/2 times occupying x/2 space, but the for loop that executes after all the recursive calls runs n times, but does not use n space. Correct?

You do not do X/2 calls, recall you are dividing X by 2 each time. The loop does not use any space. I am not sure of the space complexity because it depends on if you use the stack space used by the recursive call.

You can assume space for the recursive call is allocated on the stack and that when the recursive call returns, that space is automatically freed.