EJERCICIOS 7.25 al 7.27

//Guillermo Escobar P cod 256243

7.25. Consider the following alternative to this chapter's powerOf2 algorithm:

```
public static long otherPowerOf2( int n) {
if (n > 0) {                                        // n comparaciones
long lowerPower = otherPowerOf2( n-1);              //   n
return lowerPower + lowerPower;                     // 1
      }
     else {
     return 1;                                      //1
    }
}
```

Estimate the execution time of this algorithm.

Para un n dado su tiempo de ejecución es $2n + 2$

7.26. Redo the analysis of powerOf2's execution time, counting
1. The n > 0 comparisons as well as the additions
2. The n > 0 comparisons and the subtractions as well the additions
Do these more complete analyses still suggest execution times more or less proportional to 2n, or do the times become even worse?

En el caso 1,quedaría un tiempo de ejecución de $3n + 2$
En el caso 1,quedaría un tiempo de ejecución de $4n + 2$

De todas maneras se mantiene el orden en n, y el algoritmo tiene un orden mayor que en caso original.

7.27. What function of x does the following algorithm compute (assume x is a natural number)?

```
public static int mystery( int x) {
if (x > 0) {
return mystery( x-1) + 2* x - 1;
}
else {
return 0;
}
}
```

(Hint: The algorithm practically is a recurrence relation for the function it computes. Write this recurrence explicitly, and then find a closed form for it.).

La función mystery retorna x^2