⑤ Calcular la eficiencia del código proporcionado.

```
fun Calculo(x,y,z: entero) dev valor:entero
var i,j,t: entero
  valor ← 0
  Desde i ← x hasta y Hacer valor ← valor + i fdesde
  si (valor ÷ (x+y)) <= 1 entonces Devolver z
  si no
      t ← x + ((y-x) ÷ 2)   { ÷ es la división entera }
      Desde i ← x hasta y Hacer
            Desde j ← (3*x) hasta (3*y) Hacer
                  valor ← valor + Minimo(i,j)
            fdesde
      fdesde
      valor ← valor + 4*Calculo(t,y,valor)
      Devolver valor
  fsi
ffun
```

$\sum_{i=x}^{y} 1$ $\Big\}$ $1+1+1=3$

$\max($ 1, La otra parte es mayor

$1 + \sum_{i=x}^{y} \sum_{j=3}^{3y} 1 + \Big\}$ $3n^2$

$+ T\left(\frac{y-x}{2} + x\right) \Rightarrow T(n/2)$

$$T(n) = 1 + \sum_{i=x}^{y} 1 + \max\left(1, 1 + \sum_{i=x}^{y}\left(\sum_{j=3x}^{3y}(1)\right) + T\left(\frac{y-x}{2} + x\right)\right)$$

$T(n) = 3 + n + 3n^2 + T(n/2)$; $n = 2^K$; $T(2^K) = 3 + 2^K + 3 \cdot 2^{K^2} + T(2^{K-1})$

Homogénea asociada: $T(2^K) = 2^{K-1}$; $K-1=0$; $K=1$ (multiplicidad 1) $\quad (H) = A$

Solución particular: $(P) = BK + C2^K + D2^{K^2}$

Solución general: $(G) = A + BK + C2^K + D2^{K^2}$; $2^K = n$; $(G) = A + \log_2(n) + Cn + Dn^2$

Complejidad: $O(n^2)$

⑥ Función iterativa que determina si un número es perfecto

$T(n) = 1 + \sum_{i=0}^{n} 1 + \max(1,0) + 1$

$T(n) = 2 + 2n$

Complejidad: $O(n)$

```swift
func isPerfect (_ number:UInt) -> Bool{
    var acc :UInt = 0
    for posibleDivisor in 1..<number{
        if number % posibleDivisor == 0{
            acc += posibleDivisor
        }
    }
    return acc == number
}
```

⑨ Función recursiva para calcular el sumatorio $S = 1+2+3+\ldots+n-1+n$

$T(n) = 1 + \max(1,0) + T(n-1)$

$T(n) = 2T(n-1)$

$T(n) = A + Bn$

Complejidad: $O(n)$

```swift
func n_adder (to end:UInt, count:UInt=0, acc:UInt=0) -> UInt{
    if (end < count){
        return acc
    }
    return n_adder(to: end,count: count+1, acc: acc+count)
}
```