6. Escribí algoritmos cuyas complejidades sean (asumiendo que el lenguaje no tiene multiplicaciones ni logaritmos, o sea que no podés escribir **for** i:= 1 **to** $n^2 + 2\log n$ **do** ... **od**):

(a) $n^2 + 2\log n$ (b) $n^2 \log n$ (c) 3^n

a) **n^2 + 2 log(n)**: para obtener un algoritmo con este orden divido el problema en dos subalgoritmos por cada término del orden.

- **n^2**: este orden se consigue colocando anidando dos ciclos *for* de *1* hasta *n* de una asignación.
- **2 log(n)**: el orden log(n) se obtiene reduciendo una variable con div.

```
proc ej6a(in n: nat)
    var x,k: nat
    x := 0

    for i := 1 to n do
        for j := 1 to n do
            x := x+1
        od
    od

    k := n

    for t := 1 to 2 do
        while k > 1 do
            k := k div 2
        od
    od
end proc
```

b) **n^2 log(n)**: el ejercicio 1.b de este práctico tiene este orden.

```
proc f2(in n: nat)
    for i := 1 to n do
        for j := 1 to n do
            t := 1
        od
    od

    if n > 0 then
        for i := 1 to 4 do
            f2(n div 2)
        od
    fi
end proc
```

c) **3^n**: para obtener un orden de 3^n se puede llamar tres veces a una función e ir decreciendo su valor a medida que se la llama.

```
fun recursion(n: nat) ret res: nat
    if n ≤ 0 then
        res := n
    else
        for i := 1 to n do
            res := recursion(n-1) + recursion(n-1) + recursion(n-1)
        od
    fi
end fun
```