2. Dado un arreglo a : **array**$[1..n]$ **of nat** se define una *cima* de a como un valor k en el intervalo $1, \ldots, n$ tal que $a[1..k]$ está ordenado crecientemente y $a[k..n]$ está ordenado decrecientemente.

   (a) Escribí un algoritmo que determine si un arreglo dado tiene cima.

```
fun tiene_cima(a: array[1..n] of int) ret res: bool
    var i: nat

    i := 1
    res := true

    if n ≠ 1 then
        while i ≤ n-1 ∧ a[i] < a[i+1] do
            i := i+1
        od

        while i ≤ n-1 ∧ res do
            res := a[i] > a[i+1]
            i := i+1
        od
    fi
end fun
```

   (b) Escribí un algoritmo que encuentre la cima de un arreglo dado (asumiendo que efectivamente tiene una cima) utilizando una búsqueda secuencial, desde el comienzo del arreglo hacia el final.

```
fun cima_busqueda_secuencial(a: array[1..n] of int) ret res: nat
    var i: nat

    i := 1

    while i ≤ n-1 ∧ a[i] < a[i+1] do
        i := i+1
    od

    res := i
end fun
```

   (c) Escribí un algoritmo que resuelva el mismo problema del inciso anterior utilizando la idea de *búsqueda binaria*.

```
fun cima(a: array[1..n] of nat) ret res: nat
    res := cima_busqueda_binaria(a,1,n)
end fun

fun cima_busqueda_binaria(a: array[1..n] of nat, lft: nat, rgt: nat) ret res: nat
    var mid: nat

    if lft = rgt then
        res := lft
    else
        mid := (lft+rgt) ÷ 2
        if a[mid] < a[mid+1] → res := cima_busqueda_binaria(a,mid+1,rgt)
        □ a[mid] > a[mid+1] → res := cima_busqueda_binaria(a,lft,mid)
        fi
    fi
end fun
```

(d) Calculá y compará el orden de complejidad de ambos algoritmos.

ops(cima_busqueda_secuencial) = n, pues el peor caso es cuando a[i] < a[i+1] nunca es False, o dicho de otra forma cuando la cima se encuentra en el último elemento del arreglo.
∴ **cima_busqueda_secuencial** tiene un orden de ***complejidad lineal*** u ***O(n)***.

ops(cima_busqueda_binaria) = $log_2(n)$, pues en cada recursión se divide el tamaño del problema en dos y haciendo así menos comparaciones que la búsqueda secuencial.
∴ **cima_busqueda_binaria** tiene un orden de ***complejidad logarítmica*** u ***O(log(n))***.