4. Escribí una variante del procedimiento partition que en vez de tomar el primer elemento del segmento $a[izq, der]$ como pivot, elige el valor intermedio entre el primero, el último y el que se encuentra en medio del segmento. Es decir, si el primer valor es 4, el que se encuentra en el medio es 20 y el último es 10, el algoritmos deberá elegir como pivot al último.

```
proc variante_partition(in/out a: array[1..n] of T, in lft,rgt: nat, out ppiv: nat)
    var i,j,mid: nat

    mid := (lft+rgt) div 2

    {- busco el pivot -}
    if a[lft] ≤ a[rgt] → if a[mid] ≤ a[lft] → ppiv := lft
                         □ a[mid] ≥ a[rgt] → if a[mid] ≤ a[rgt] → ppiv := rgt
                                             □ a[mid] ≥ a[rgt] → ppiv := mid
                                             fi
                         fi
    □ a[lft] ≥ a[rgt] → if a[mid] ≥ a[lft] → ppiv := lft
                         □ a[mid] ≤ a[lft] → if a[mid] ≥ a[rgt] → ppiv := mid
                                             □ a[mid] ≤ a[rgt] → ppiv := rgt
                                             fi
                         fi
    fi

    if ppiv = lft → i := lft+1
                    j := rgt
                    while i ≤ j do
                        if a[i] ≤ a[ppiv] → i := i+1
                        □ a[j] ≥ a[ppiv] → j := j-1
                        □ a[i] > a[ppiv] ∧ a[j] < a[ppiv] → swap(a,i,j)
                                                            i := i+1
                                                            j := j-1
                        fi
                    od
                    swap(a,j,ppiv)
    □ ppiv = rgt → i := lft
                   j := rgt-1
                   while i ≤ j do
                       if a[i] ≤ a[ppiv] → i := i+1
                           □ a[j] ≥ a[ppiv] → j := j-1
                           □ a[i] > a[ppiv] ∧ a[j] < a[ppiv] → swap(a,i,j)
                                                               i := i+1
                                                               j := j-1
                       fi
                   od
                   swap(a,j,ppiv)
    □ ppiv = mid → i := lft
                   j := rgt
                   while i ≤ j do
                       if a[i] ≤ a[ppiv] → i := i+1
                       □ a[j] ≥ a[ppiv] → j := j-1
                       □ a[i] > a[ppiv] ∧ a[j] < a[ppiv] → swap(a,i,j)
                                                           i := i+1
                                                           j := j-1
                       fi
                   od
                   swap(a,j,ppiv)
    fi
end proc
```