Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

1) **Problema 1 (5 puntos)**. Sean A y B dos vectores de *N* elementos enteros, ordenados circularmente y que pueden contener números repetidos. Ambos vectores comparten exactamente los mismos elementos hasta una posición determinada 'k', **a partir de la cual TODOS sus elementos serán diferentes**. Se pide implementar un algoritmo, que dado los vectores A y B determine esa posición 'k'. En el caso de que los dos vectores sean idénticos el procedimiento devolverá -1 (indicando de esa forma que tal posición no existe).

   a) **(3.5 puntos)** Diseñar el procedimiento basado en Divide y Vencerás con complejidad O(log N) en el caso peor[1] (donde N es el tamaño del vector) que devuelva un número entero que corresponde a la posición del primer elemento diferente entre ambos vectores.

   Ejemplo:

0	1	2	3	4	5	6	7	8
-4	-2	0	1	1	2	3	4	-9

0	1	2	3	4	5	6	7	8
-4	-2	0	1	1	5	7	10	-6

```
int ejDyVsept (int[] vector1, int[] vector2)

Posicion = 5
```

```
Se utiliza una función auxiliar para buscar en cada una de las partes
del vector.
La longitud de los vectores es la misma, el hecho del orden circular
no afecta.

public static void main(String[] args) {
     //Posicion =4
     int[] v1 = {-2,0,1,1,2,2,2,3};
     int[] v2 = {-2,0,1,1,5,5,5,5};
     // Posicion =3
     //int[] v1 = {-9,-5,-2,1,1,2,3,4,-10,-12,-14,-16};
     //int[] v2 = {-9,-5,-2,-1,-1,5,7,10,-11,-13,-15,-17};
     //Posicion =5
     //int[] v1 = {-4,-2,0,1,1,2,3,4,-9};
     //int[] v2 = {-4,-2,0,1,1,5,7,10,-6};

    System.out.print("v1 = ");
    for (int i = 0; i < v1.length; i++)
        System.out.print(v1[i] + "  ");
    System.out.println();
```

[1] Desarrollar un algoritmo que tenga una complejidad diferente a O(log N) en el caso peor conllevará una puntuación de 0 en la pregunta.

```
    System.out.print("v2 = ");
    for (int i = 0; i < v2.length; i++)
        System.out.print(v2[i] + "  ");
    System.out.println();
    System.out.println("Posicion: "+ejDyVsept(v1,v2));

}

private static int ejDyVseptAux(int v1[], int[] v2, int i0, int iN) {
    boolean encontrado = false;
    if (i0==iN) { //O(1)
        if (v1[i0] == v2[i0])
                return -1;
        else
            return i0;
    } else {
        int k = (i0 + iN) / 2; //O(1)
        //O(T(N/2) N>1, 2º caso TM, Log(N)
        if(v1[k] == v2[k]) //busco lado derecho
            return ejDyVseptAux(v1, v2, k+1, iN);
        else //busco lado derecho;
            return ejDyVseptAux(v1, v2, i0, k);
    }
}

 private static int ejDyVsept(int[] v1,int[] v2) {
        return ejDyVseptAux(v1, v2, 0, v1.length-1);
    }
```

b) (1.5 puntos) Justifica que la complejidad del algoritmo desarrollado en el apartado anterior para el caso peor es O(log *N*).

La condición de orden circular no afecta al procedimiento, únicamente se trata de encontrar el primer elemento diferente, en el caso de que este exista (es decir, cuando no son dos vectores iguales). Si aplicamos el TM a la función auxiliar (2º caso del TM) nos genera una complejidad O(log N).

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ T\left(\frac{N}{2}\right) + \Theta(1) & N > 1 \end{cases}$$

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 1 \cdot T\left(\frac{N}{2}\right) + \Theta(N^0) & N > 1 \end{cases}$$

Teorema Maestro:

(2º Caso)
$\text{Log}_2(1)=0$

Complejidad: $\Theta(\log N)$