Problema 1 (6 puntos)

Una de las limitaciones que tienen los arrays en Java es que, una vez creados, no podemos cambiar su tamaño. En este problema definiremos, entre otras, una clase cuyo propósito es almacenar tantos elementos como sea necesario.

Apartado a: la clase Interval (2 puntos)

Diseñad e implementad la siguiente clase para representar intervalos entre un mínimo y un máximo. La clase tendrá los métodos públicos:

- `new Interval(int min, int max)`
  - crea un intervalo con los límites dados
  - en ningún momento se comprobará que `min` sea menor o igual que `max`
- `boolean contains(int elem)`
  - devuelve true si **`min ≤ elem < max`**; false en caso contrario
- `boolean isEqual(Interval other)`
  - devuelve true si el intervalo tiene los mismos límites que other; false en caso contrario

Apartado b: la clase NonEmptyIntervalSet (4 puntos)

Se desea tener una colección ampliable de intervalos, es decir, una clase en la que se puedan añadir intervalos, tantos como se quiera.
Esta clase se llamará NonEmptyIntervalSet y tendrá los siguientes métodos públicos:

- `new NonEmptyIntervalSet()`
  - crea un conjunto vacío para guardar intervalos
  - inicialmente se reservará un espacio para poder guardar 10 intervalos.
- `boolean addInterval(int min, int max)`
  - si el intervalo es vacío (**`max ≤ min`**), no hace nada y devuelve falso
  - si ya hay un intervalo con los mismos límites en la colección, no hace nada y devuelve falso
  - si no pasa nada de eso, se añade el intervalo a la colección y se devuelve cierto
    - en caso de no haber espacio ya, se deberá doblar la capacidad para poder añadir el nuevo intervalo
- `int numContains(int elem)`
  - devuelve el número de intervalos del conjunto que contienen al número `elem`

Problema 2 (4 puntos)

Diseñad un programa principal tal que

- pida al usuario una línea con los mínimos
- pida al usuario una línea con los máximos
  - ambas cadenas contienen números enteros (pueden ser negativos) separados por espacios

  - ambas cadenas contienen por lo menos un valor, pero el número de valores contenidos puede ser diferente
  - se consideran los intervalos correspondientes a las parejas de mínimos y máximos, es decir, el primer mínimo con el primer máximo, el segundo mínimo con el segundo máximo, etc., etc. y que ni están vacíos ni son iguales a intervalos anteriores
- pida al usuario una línea con posibles valores
  - la línea consiste en números enteros (pueden ser negativos) separados por espacios
  - la línea tiene como mínimo un valor
- escriba el valor que pertenece a más intervalos y el número de intervalos a los que pertenece
  - en caso de que haya varios valores posibles, podéis escribir cualquiera de ellos
  - si ningún número pertenece a ningún intervalo, el valor máximo pertenece a 0 intervalos

Ejemplo de uso:

```
Entra los mínimos:  3 6 8 6 5 -3 4 -1
Entra los máximos: 14 8 9 8 9 14 9 10 8 15 2
Entra los números: 4 8 3 9
El máximo es 8 y pertenece a 6 intervalos
```

- class **CommandLineProgram**
  - String **readLine**(String message)
  - void **println**(String str) / void **println**(int n) / **println**(double d)
  - void **print**(String str) / void **print**(int n) / **print**(double d)
- class **Double**
  - static double **parseDouble**(String str)
- class **Integer:**
  - static int **parseInt**(String str)
- class **String:**
  - char **charAt**(int index)
  - int **length**()
  - boolean **equals**(String other) / boolean **equalsIgnoreCase**(String other)
  - int **compareTo**(String other)
  - int **indexOf**(char c) / int **indexOf**(String s)
  - String **substring**(int p1, int p2) / String **substring**(int p1)
  - String **concat**(String s) / o usar + para concatenar Strings
  - String **trim**()
  - static String **valueOf**(int n)
  - static String **valueOf**(double d)
- class **StringTokenizer:**
  - new **StringTokenizer**(String str)
  - new **StringTokenizer**(String str, String delims)
  - new **StringTokenizer**(String str, String delims, boolean returnDelims)
  - boolean **hasMoreTokens**()
  - String **nextToken**()
- class **Math**
  - static int **max**(int n1, int n2) / static int **min**(int n1, int n2)
  - static double **max**(double d1, double d2)
  - static double **min**(double d1, double d2)