ENA VERHORST
7/11/19

2. Ordenación por raíz (1,5 puntos)

Se pide implementar el algoritmo de ordenación por raíz:

```
public void ordRaiz(String[] a, int maxLetras)
// Pre: maxLetras indica el número de letras de los valores a ordenar
//      todas las letras son minúsculas (entre 'a' y 'z')
// Post: a está ordenado ascendentemente (método de la raíz)
```

Ejemplo

- Palabras de 3 letras
- Secuencia desordenada: "aye", "yea", "aya", "yay", "eda", "aca", "bea", "cae"
- Fase número 1:
  - Las vamos colocando en colas, según su letra menos significativa

0 (a)	"yea" "aya" "eda" "aca" "bea"
1 (b)	
2 (c)	
3 (d)	
4 (e)	"aye" "cae"
5	
6	
...	
24 (y)	"yay"
25 (z)	

  - Las volvemos a juntar en una cola, comenzando por la cola 0 y terminando por la cola 25:
    - "yea" "aya" "eda" "aca" "bea" "aye" "cae" "yay"

- Fase número 2:
  - Las vamos colocando en colas, según su segunda letra menos significativa

0 (a)	"cae" "yay"
1 (b)	
2 (c)	"aca"
3 (d)	"eda"
4 (e)	"yea" "bea"
5	
6	
...	
24 (y)	"aya" "aye"
25 (z)	

- Las volvemos a juntar en una cola, comenzando por la cola 0 y terminando por la cola 25:
  - "cae" "yay" "aca" "eda" "yea" "bea" "aya" "aye"

- Fase número 3: Las vamos colocando en colas, según su tercera letra menos significativa

0 (a)	"aca" "aya" "aye"
1 (b)	"bea"
2 (c)	"cae"
3 (d)	
4 (e)	"eda"
5	
6	
...	
24 (y)	"yay" "yea"
25 (z)	

- Las volvemos a juntar en una cola, comenzando por la cola 0 y terminando por la cola 25:
  - "aca" "aya" "aye" "bea" "cae" "eda" "yay" "yea"
  - ¡Ya están ordenadas!

Se pide:

- Implementar el algoritmo
- Calcular su coste de manera razonada.

Nota: Se puede hacer uso de la siguiente función:

```
public int posChar(char c)
 // Pre: c es una letra entre 'a' y 'z'
 // Post: el resultado es un valor entre 0 y 25 correspondiente a
 //       la posición de la letra en el alfabeto
 //       ('a' en la posición 0, … 'z' en la 25)
```