Partición de una lista doblemante enlazada

Queremos extender *mediante herencia* la clase `double_linked_list_ed<int>`, que implementa las colas dobles de enteros mediante listas de nodos dinámicos, doblemente enlazados, circular y con nodo fantasma, con un nuevo método

```
void particion(int pivote);
```

Este método recibe un entero `pivote` y reorganiza los nodos dinámicos de la lista enlazada de tal forma que al principio aparezcan aquellos que contengan enteros menores o iguales que el pivote y al final aparezcan los que contengan enteros mayores que el pivote.

Por ejemplo, si la lista contiene los valores `5 10 4 7 9 3` y se invoca el método `particion(8)`, la lista se transforma en `5 4 7 3 10 9`.

Observa que las posiciones relativas entre los elementos de las dos partes es la misma antes y después de la reorganización. Por ejemplo, como el `5` aparece antes que el `4` en la lista original y ambos son menores que el pivote, en la lista resultado aparecen al principio también así, el `5` antes que el `4`.

En la implementación del método no pueden hacerse nuevos `new`s, ni *copiar* los enteros de un nodo a otro. También hará falta un método `print` para mostrar por pantalla los elementos de la lista, de inicio a fin.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en dos líneas. La primera contiene dos números: el número N de elementos de la lista (un número entre 1 y 100.000) y el valor del `pivote`. En la segunda se muestran los N elementos de la lista, números enteros entre 1 y 1.000.000.

Salida

Para cada caso de prueba se escribirá en una línea la lista modificada tras reorganizar sus elementos según el valor del pivote.

Entrada de ejemplo

```
6 8
5 10 4 7 9 3
7 4
1 2 3 4 5 6 7
7 4
7 6 5 4 3 2 1
```

Salida de ejemplo

```
5 4 7 3 10 9
1 2 3 4 5 6 7
4 3 2 1 7 6 5
```

Autor: Alberto Verdejo e Isabel Pita