Ejercicio 1. Operación `partition` en TAD `list` (3 puntos)

Se pide extender el TAD `list` con una nueva operación `void partition(int pivote);` que 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 partition(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 puede crearse ni destruirse memoria dinámica ni copiar los enteros de un nodo a otro. Indica y justifica la complejidad del método implementado.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso consiste en dos líneas. La primera contiene dos números: el número *N* de elementos de la lista y el valor del pivote. En la segunda aparecen los *N* elementos de la lista separados por espacios.

Salida

Para cada caso de prueba se escribirán en una línea los elementos de la lista modificada separados por espacios.

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
```