Transposición

"Transponer" una lista con respecto a un valor de transposición *v* consiste en intercambiar el elemento que va inmediatamente antes de la primera aparición de *v* en la lista por el elemento que va inmediatamente después de dicha primera aparición. Por ejemplo, el resultado de "transponer" la lista

0	1	2	3	2	5	6	7

con respecto a 2 es

0	3	2	1	2	5	6	7

Si alguno de los elementos involucrados en el intercambio no existe, la operación no tendrá ningún efecto.

Añade una nueva operación mutadora

```
void transponer(const T& val);
```

a la implementación del TAD `Lista` basada en nodos doblemente enlazados, que "transponga" la lista con respecto al valor `val`, y determina justificadamente su complejidad. Dicha operación no puede invocar, ni directa ni indirectamente, operaciones de manejo de memoria dinámica (`new`, `delete`), ni tampoco puede realizar asignaciones a los contenidos de los nodos.

Nota:

La implementación debe realizarse sobre el archivo `lista.h` que se proporciona con el código de apoyo, ya que dicho archivo incluye un par de métodos adicionales que se utilizan en las pruebas.

Se proporciona también un archivo `main` que está preparado para funcionar en dos modos:

- En modo de *ejecución*. En este modo, el programa lee líneas que representan listas de enteros y valores de transposición, y escribe las listas transpuestas con respecto a los valores dados, y la inversa de dichas transposiciones. Más concretamente:
  - El primer valor de la línea leída representa el número de elementos en la lista. El segundo valor el valor de transposición *v*. El resto de valores los elementos de la lista a transponer. Por ejemplo, la línea `5 0 1 2 0 3 4` indica que (i) se desea transponer una lista de 5 elementos; (ii) el valor de transposición es 0; (iii) los elementos de la lista son `1 2 0 3 4`.
  - En la salida se muestran los elementos de la lista transpuesta, seguidos de #, seguida de los elementos de la lista transpuesta, pero listados al revés (primero el último, después el penúltimo, etc). Por ejemplo, dada la línea anterior, se imprimirá `1 3 0 2 4#4 2 0 3 1`.

  El final de la entrada se indica con una línea que contiene únicamente -1.

  Ejemplo de entrada / salida:

Entrada	**Salida**
5 0 1 2 0 3 4 7 0 1 2 0 3 0 5 6 3 0 5 0 6 3 0 0 5 6 3 0 5 6 0 -1	1 3 0 2 4#4 2 0 3 1 1 3 0 2 0 5 6#6 5 0 2 0 3 1 6 0 5#5 0 6 0 5 6#6 5 0 5 6 0#0 6 5

- En modo de *autocorrección*. En este modo ejecuta un conjunto de casos de prueba, y comprueba que la implementación se ajusta a dichos casos de prueba. **Importante**: que el programa supere satisfactoriamente todos los casos de prueba **no significa** que la implementación sea correcta; sólo significa que funciona para los casos de prueba proporcionados.

Para que el programa se ejecute en modo *autocorrección* basta con descomentar

```
#define AUTO
```

al comienzo de `main.cpp`. Por lo demás, `main.cpp` no debe modificarse.