Desentramar una lista doblemente enlazada

Desentramar una lista consiste en separar, por un lado, los elementos que están en las posiciones pares y, por otro lado, los que están en posiciones impares. Por ejemplo, tras desentramar la lista `[10, 9, 3, 8, 2, 7, 5]` se obtienen como resultado las listas `[10, 3, 2, 5]` y `[9, 8, 7]`.

En este ejercicio partimos de la clase `ListLinkedDouble`, que implementa el TAD lista mediante listas doblemente enlazadas circulares con nodo fantasma. Queremos añadir un nuevo método, llamado `unzip()`:

```
class ListLinkedDouble {
private:
  struct Node {
    T value;
    Node *next;
    Node *prev;
  };
  Node *head;
  int num_elems;

public:
  ...
  void unzip(ListLinkedDouble & other);
};
```

El nuevo método desentrama la lista `this`, quedándose esta última con los elementos situados en posiciones pares (suponemos que las posiciones se empiezan a numerar desde el 0), y moviendo los de las posiciones impares al final de la lista `other` pasada como parámetro. Por ejemplo, dada la lista `xs = [10, 1, 20, 2, 30, 3]`, y la lista `zs = []`, tras la llamada `xs.unzip(zs)` la lista `xs` tiene los valores `[10, 20, 30]` y la lista `zs` tiene los valores `[1, 2, 3]`.

Si, en el ejemplo anterior, la lista `zs` no estuviese vacía antes de la llamada a `unzip`, se añadirían los elementos de las posiciones impares de `xs` al *final* de la lista `zs`. Por ejemplo, si inicialmente tuviésemos `zs = [5, 0]`, tras hacer `xs.unzip(zs)` tendríamos `zs = [5, 0, 1, 2, 3]`.

Se pide:

1. Implementar el método `unzip()`.
2. Indicar su coste con respecto al tamaño de la lista de entrada.

Importante: Para la implementación del método no pueden crearse, directa o indirectamente, nuevos nodos mediante `new` ni borrar nodos mediante `delete`; han de reutilizarse los nodos de las listas de entrada. Tampoco se permite copiar valores de un nodo a otro. El coste de la operación ha de ser lineal con respecto a la longitud de ambas listas.

Entrada

La entrada comienza con un número que indica el número de casos de prueba que vienen a continuación. Cada caso de prueba consiste en dos líneas: la primera de ellas denota la lista `this`, y la

segunda denota la lista `other`. Cada una de las listas se representa mediante una secuencia con sus elementos (números enteros distintos de cero). Cada secuencia finaliza con `0`, que no forma parte de la lista.

Salida

Para cada caso de prueba se imprimirán dos líneas: una con el contenido de la lista `this` tras llamar al método `unzip` y otra con el contenido de la lista `other` tras esa misma llamada. Para imprimir las listas puedes utilizar el método `display()`, o la sobrecarga del operador << que se proporciona para listas.

Entrada de ejemplo

```
2
10 1 20 2 30 3 0
0
7 6 5 4 3 2 1 0
10 20 0
```

Salida de ejemplo

```
[10, 20, 30]
[1, 2, 3]
[7, 5, 3, 1]
[10, 20, 6, 4, 2]
```

Autor

Manuel Montenegro