Entramar dos listas doblemente enlazadas

Entramar dos listas `xs` y `zs` consiste en fusionarlas de modo que los elementos de `zs` se intercalan con los de `xs`. Por ejemplo, al entramar las listas `[10, 20, 30]` y `[4, 7, 9]` se obtiene como resultado `[10, 4, 20, 7, 30, 9]`. En el caso en que una lista sea más larga que la otra, se colocan los elementos sobrantes al final. Por ejemplo, al entramar `[1, 2]` con `[6, 7, 8, 9]` se obtiene `[1, 6, 2, 7, 8, 9]`.

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 `zip()`:

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

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

El método `zip()` entrama las listas `this` y `other`, dejando el resultado en `this`, y dejando la lista `other` vacía. Por ejemplo, si xs = `[1, 2, 3]` y `zs = [10, 20, 30]`, tras hacer `xs.zip(zs)` tenemos que `xs = [1, 10, 2, 20, 3, 30]` y `zs = []`.

Se pide:

1. Implementar el método `zip()`.
2. Indicar su coste con respecto al tamaño de las listas 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 cuatro líneas. La primera línea contiene un número N indicando cuántos elementos tiene la lista `this`. La segunda línea contiene esos N elementos, separados por espacios. La tercera línea contiene un número M indicando cuántos elementos tiene la lista `other`. La cuarta línea contiene esos M elementos, separados por espacios. Puedes suponer que los elementos de ambas listas están en orden creciente, y que ninguna lista tiene elementos duplicados.

Salida

Para cada caso de prueba se imprimirá el contenido de la lista `this` tras llamar al método `intersect()`. Puedes utilizar el método `display()` de esta clase.

Entrada de ejemplo

```
3
4
1 3 5 7
4
2 4 6 8
3
7 3 9
0

2
2 1
4
5 3 4 6
```

Salida de ejemplo

```
[1, 2, 3, 4, 5, 6, 7, 8]
[7, 3, 9]
[2, 5, 1, 3, 4, 6]
```

Autores

Manuel Montenegro y Alberto Verdejo