Desentramar una lista enlazada simple

Desentramar una lista consiste en separar, por un lado, los elementos que están en las posiciones pares y, por otro lado, los que estan 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]`.

Partimos de la clase `ListLinkedSingle<T>`, que implementa el TAD lista mediante listas enlazadas simples con nodo fantasma, un puntero adicional (`last`) al final de la lista y un contador con el número de elementos. Queremos añadir un nuevo método, llamado `unzip()`:

```
template<typename T>
class ListLinkedSingle {
private:
  struct Node {
    T value;
    Node *next;
  };
  Node *head;
  Node *last;
  int num_elems;

public:
  ...
  void unzip(ListLinkedSingle &dest);
};
```

El método `unzip()` desentrama la lista `this`, quedándose solamente con los elementos situados en posiciones pares (suponemos que las posiciones se empiezan a contar desde el 0), y moviendo los de las posiciones impares al final de la lista `dest` 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]`.

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 la lista de entrada. Tampoco se permite copiar valores de un nodo a otro. El coste de la operación ha de ser lineal con respecto a `this.num_elems`.

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 `dest`. La cuarta línea contiene esos M elementos, separados por espacios.

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 `dest` 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
6
10 1 20 2 30 3
0

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

Salida de ejemplo

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