Rellenar los huecos de una lista doblemente enlazada

Rellenar los huecos de una lista de enteros consiste en insertar elementos en la misma de modo que la diferencia entre dos elementos consecutivos sea igual o menor que uno. Por ejemplo, al rellenar los huecos de la lista `[1, 4, 2, 6]` se obtiene como resultado `[1, 2, 3, 4, 3, 2, 3, 4, 5, 6]`, donde los elementos insertados se muestran en color rojo.

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

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

public:
  ...
  void fill_gaps(int pos_begin, int pos_end);
};
```

Este método rellena los huecos del segmento de lista comprendido entre las posiciones `pos_begin` y `pos_end`, ambas posiciones incluidas. Puedes suponer que $0 \leq$ `pos_begin` $\leq$ `pos_end` $< N$, donde N es el número de elementos de la lista. El primer elemento de la lista está en la posición 0.

Por ejemplo, dada la lista `xs = [1, 6, 9, 4, 7, 3, 5, 8]`, y los números `pos_begin = 1` y `pos_end = 5`, el segmento delimitado por ambos extremos se marca a continuación:

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

Tras la llamada `xs.fill_gaps(1, 5)`, `xs` queda del siguiente modo:

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

Importante: Para la implementación del método puedes utilizar `new` solamente para crear los nodos con los elementos que se añadan a la lista (esto es, los de color rojo). Con respecto a los restantes, has de reutilizar los de la lista `this` original. No se pide indicar el coste de la operación.

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 línea contiene tres números N, i y j tales que $0 \leq i \leq j < N$, y donde N es el número de elementos de la lista, e i y j son las posiciones inicial y final del segmento sobre el que se aplicará el método `fill_gaps()`. La segunda línea contiene los N elementos de la lista, separados por espacios.

Salida

Para cada caso de prueba se imprimirá el contenido de la lista `this` tras llamar al método `fill_gaps()`. Puedes utilizar el método `display()` de la clase, o directamente la sobrecarga del operador << que se proporciona con la clase.

Entrada de ejemplo

```
3
8 1 5
1 6 9 4 7 3 5 8
5 0 3
1 2 3 2 1
2 0 1
3 10
```

Salida de ejemplo

```
[1, 6, 7, 8, 9, 8, 7, 6, 5, 4, 5, 6, 7, 6, 5, 4, 3, 5, 8]
[1, 2, 3, 2, 1]
[3, 4, 5, 6, 7, 8, 9, 10]
```