Repetir los elementos de una lista

Dada una lista `xs` de números enteros, queremos repetir cada elemento de `xs` tantas veces como indique otra lista `ys`. Dicho de otro modo, si `xs` e `ys` son dos listas de igual longitud N, entonces para cada i tal que $0 \leq i < N$:

- Si `ys[i] = 0`, entonces `xs[i]` se elimina de la lista `xs`.
- Si `ys[i] = 1`, entonces `xs[i]` se queda tal y como está.
- Si `ys[i] = v`, donde $v > 1$, entonces `xs[i]` se queda tal y como está, y además se añaden $n-1$ copias del elemento `xs[i]` tras la posicion i-ésima.

La lista `ys` recibe el nombre de *lista de multiplicidades*.

Por ejemplo, dadas las siguientes listas `xs` e `ys`,

```
xs  =  [23, 10, 5, 9]
ys  =  [ 2,  0, 1, 3]
```

la lista resultado debe ser `[23, 23, 5, 9, 9, 9]`. Es decir, el número 23 aparece dos veces en el resultado, el número 10 aparece 0 veces (es decir, se elimina de la lista `xs`), el número 5 aparece una vez, y el número 9 aparece tres veces.

Partimos de la clase `ListLinkedSingle`, que implementa el TAD lista utilizando listas enlazadas simples:

```
class ListLinkedSingle {
private:
  struct Node {
    int value;
    Node *next;
  };

  Node *head;
public:
  ...

  // Nuevo metodo
  void replicate(const ListLinkedSingle &ys);
};
```

El método `replicate` recibe una lista de multiplicidades `ys` y modifica la lista `this` preservando, repitiendo, o borrando cada elemento según indique su multiplicidad correspondiente en `ys`. En particular, si `xs` e `ys` son las listas del ejemplo anterior, tras la llamada `xs.replicate(ys)`, la variable `xs` debe representar la lista `[23, 23, 5, 9, 9, 9]`. La lista `ys` no debe modificarse.

En esta práctica se pide:

1. Implementar el método `replicate`. Para ello deben crearse la cantidad mínima necesaria de nodos para construir el resultado. Es decir, si un elemento de la lista `this` ha de tener multiplicidad distinta de cero, el nodo que contenga ese elemento ha de formar parte de la lista resultado. No se permite cambiar el valor contenido en los nodos ya existentes en la lista `this`.

2. Indicar el coste en tiempo del método `replicate`.
3. Escribe un programa que lea de la entrada varios casos de prueba. Cada caso de prueba contiene dos listas: la lista `xs` sobre la que actuará el método `replicate` y la lista de multiplicidades `ys` pasada como parámetro. El programa de prueba debe imprimir el resultado de `xs.replicate(ys)`. Ten en cuenta que los elementos de las listas de entrada están en orden inverso: si la entrada es `9 5 10 23` debes construir la lista `[23, 10, 5, 9]`. Para construir las listas utiliza el método `push_front`.

Entrada

Cada entrada comienza con un número indicando cuántos casos de prueba vienen a continuación.

Cada caso de prueba ocupa tres líneas. La primera linea contiene un número N con la longitud de ambas listas ($0 \leq N \leq 50000$). La segunda línea contiene N números con los elementos de la lista `xs` sobre la que se aplicará el método `replicate`. La tercera línea contiene N números con las multiplicidades, que son números menores que 50000. Cada una de estas dos últimas líneas contiene los elementos de sendas listas **en orden inverso**.

Salida

Para cada caso de prueba ha de imprimirse una línea con los elementos de la lista `xs` tras aplicar el método `replicate`. Puedes utilizar, para ello, el método `display` de las listas. Se garantiza que la lista resultado tiene longitud inferior a 10^5.

Entrada de ejemplo

```
3
4
9 5 10 23
3 1 0 2
3
1 4 3
5 1 5
3
2 6 1
0 0 0
```

Salida de ejemplo

```
[23, 23, 5, 9, 9, 9]
[3, 3, 3, 3, 3, 4, 1, 1, 1, 1, 1]
[]
```

Créditos

Manuel Montenegro