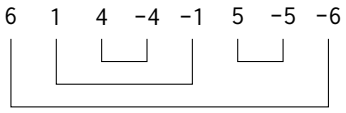
Reconstruir una lista equilibrada

Supongamos una lista de números enteros, todos distintos entre sí y todos distintos de `0`, tal que para cada número x de la lista, su opuesto $-x$ también está en la lista. En este caso, decimos que x y $-x$ están *emparejados*. Una lista de este tipo está *equilibrada* si, para cada pareja $(x, -x)$ de la lista, el número positivo de la pareja siempre aparece antes que el negativo y además las parejas están correctamente anidadas unas dentro de otras. Es decir, para cada número negativo de la lista, su pareja es el número opuesto más cercano (por la izquierda) que aún no estaba emparejado con otro. Por ejemplo, la lista `10 6 -6 -10` está equilibrada, pero la lista `10 6 -10 -6` no lo está:

Otro ejemplo de lista equilibrada es `6 1 4 -4 -1 5 -5 -6`:



Tenía varias secuencias equilibradas guardadas en un fichero, pero resulta que el fichero se ha corrompido de una manera un tanto extraña, porque solo se han visto afectados los números negativos. Algunos de los que estaban al final de la lista han desaparecido sin más. Otros han sido reemplazados por un `0`, y por si fuera poco, han aparecido números negativos que no estaban en la lista original. Por ejemplo, al intentar recuperar la lista anterior me he encontrado con lo siguiente:

```
6 1 4 0 0 5 -7 -5
```

Como puedes ver, el número `-6`, que estaba al final de la lista, ha desaparecido. Los números `-4` y `-1` de la lista original han sido reemplazados por ceros. Además ha aparecido el número `-7`, que no estaba en la lista original.

Pese al fastidio, creo que he tenido suerte, porque puedo reconstruir la lista original a partir de la lista corrupta.

En este ejercicio se pide implementar una función con la siguiente cabecera:

```
void reconstruir(list<int> &lista);
```

La función recibe una lista corrupta, y debe transformarla para obtener la lista equilibrada original. Para ello se deben aplicar los cambios **directamente sobre la `lista` pasada como parámetro**, pudiéndose eliminar, añadir o modificar elementos de ella. Se permite el uso de otros TADs auxiliares, pero **no** se permite construir el resultado en otra lista paralela, para luego volcarlo a la lista pasada como parámetro.

No olvides indicar el coste de la función `reconstruir`.

Entrada

La entrada consta de varios casos de prueba. Cada uno de ellos ocupa dos líneas. La primera línea contiene el número *N* de elementos de la lista corrupta. La segunda lista contiene los *N* elementos de la lista corrupta, separados por espacios.

Salida

Para cada caso de prueba ha de imprimirse una línea con los elementos de la lista equilibrada original, separados por espacios. Si la lista original es vacía, debe imprimirse una línea en blanco.

Entrada de ejemplo

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

Salida de ejemplo

```
6 1 4 -4 -1 5 -5 -6
4 5 -5 3 -3 -4

1 -1
```