## Afegir els propis elements d'una llista al final i en ordre invers (Pro2) X65021_ca

Heu d'implementar una funció que rep una llista d'enters $[x_0, x_1, x_2, \ldots, x_{n-2}, x_{n-1}]$ com a paràmetre per referència. La funció haurà d'insertar, al final de la llista, els elements que contenia inicialment però en ordre invers. És a dir, la funció retorna la llista:

$$[x_0, x_1, x_2, \ldots, x_{n-2}, x_{n-1}, x_{n-1}, x_{n-2}, \ldots, x_2, x_1, x_0]$$

**Important:** heu de garantir que els elements que la llista contenia inicialment han de quedar inalterats i ocupant les posicions del principi. En particular, la funció no els pot eliminar i tornar a afegir després.
Aquesta és la capcelera:

```
// Pre: Sigui [x0,x1,x2,...,x{n-1}] el valor inicial de l.
// Post: El valor de l és [x0,x1,x2,...,x{n-1},x{n-1},...,x2,x1,x0].
//       A més a més, els elements inicials de la llista han persistit i
//       no han canviat de valor, i ocupen les posicions del principi.
void appendReverseOrder(list<int> &l);
```

Aquí tenim un exemple de comportament de la funció:

```
appendReverseOrder(L = [2,3,1]) => L = [2,3,1,1,3,2]
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `main.cc`, `appendReverseOrder.hh`. Us falta crear el fitxer `appendReverseOrder.cc` amb els corresponents `includes` i implementar-hi la funció anterior. Només cal que pugeu `appendReverseOrder.cc` al jutge.

### Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en una llista d'enters en una línia. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

### Sortida

Per a cada cas, la sortida conté el corresponent resultat de la funció. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

### Exemple d'entrada 1

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

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

### Exemple de sortida 1

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

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

### Observació

La vostra funció i subfuncions que creeu han de treballar només amb llistes. Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, de cost lineal i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.
Una solució que elimina els elements originals de la llista i els torna a afegir més tard rebrà un 0.

### Informació del problema

Autoria: PRO2

Generació: 2026-01-25T17:30:20.537Z