Inversió dels elements d'una llista doblement encadenada, circular i amb fantasma X90856_ca

Donada la classe *Llista* que permet guardar seqüències d'enters amb una llista doblement encadenada, circular i amb fantasma, cal implementar el mètode

void *inverteix* ()

que inverteix els elements del paràmetre implícit. No es permet usar estructures auxiliars per invertir els elements ni modificar el camp info, només modificar els encadenaments dels nodes.
Cal enviar a jutge.org només la implementació del mètode *inverteix*. Al principi del mètode implementat, dins d'un comentari, cal indicar el cost temporal amb el raonament corresponent, incloent l'equació de la recurrència si fos necessari. La classe *Llista* té la següent especificació:

```
#include <vector>
#include <cstddef>
using namespace std;
typedef unsigned int nat;

class Llista {
  // Llista doblement encadenada, circular i amb fantasma.
  private:
    struct node {
      int info; // Informació del node
      node *seg; // Punter al següent element
      node *ant; // Punter a l'anterior element
    };
    node *_prim; // Punter a l'element fantasma
    nat _long;   // Nombre d'elements

  public:
    Llista ();
    // Pre: True
    // Post: El p.i. és una llista buida.

    Llista (const vector<int> &v);
    // Pre: True
    // Post: El p.i. conté els elements de v amb el mateix ordre.

    ~Llista ();
    // Post: Destrueix els elements del p.i.

    nat longitud() const;
    // Pre: True
    // Post: Retorna el nombre d'elements del p.i.

    void mostra() const;
```

```
    // Pre: True
    // Post: Mostra el p.i. pel canal estàndard de sortida.

    void mostra_invertida () const;
    // Pre: True
    // Post: Mostra el p.i. en ordre invers pel canal estàndard de sortida.

    void inverteix ();
    // Pre: True
    // Post: S'ha invertit l'ordre els elements del p.i.
    // Exemple: [2 5 3] quedaria [3 5 2]
    // No es permet usar estructures auxiliars per invertir
    // els elements ni modificar el camp info,
    // només modificar els encadenaments dels nodes.
};
```

Per testejar la solució, jutge.org ja té implementats la resta de mètodes de la classe *Llista* i un programa principal que processa línies d'enters amb els que crea llistes i després crida el mètode *inverteix*.

Entrada

L'entrada conté vàries línies formades per seqüències d'enters. Cadascuna d'elles són els elements que tindrà cada llista.

Sortida

Per a cada línia d'entrada, escriu una línia amb el resultat després d'haver invertit els elements de la llista: El nombre d'elements de la llista seguit d'un espai, els elements de la llista entre claudàtors i separats per espais, i finalment aquests mateixos elements però amb ordre invers, també entre claudàtors i separats per espais.

Observació

Cal enviar la solució (el fitxer *solution.cpp*) comprimida en un fitxer *.tar*:

```
tar cvf solution.tar solution.cpp
```

Només cal enviar la implementació del mètode *inverteix*. Seguiu estrictament la definició de la classe de l'enunciat.
Al principi del mètode implementat, dins d'un comentari, cal indicar el cost temporal amb el raonament corresponent, incloent l'equació de la recurrència si fos necessari.

Exemple d'entrada 1

```
3 -6 8 0 4 -2
```

Exemple de sortida 1

```
6 [-2 4 0 8 -6 3] [3 -6 8 0 4 -2]
```

Exemple d'entrada 2

Exemple de sortida 2

```
0 [] []
```

Exemple d'entrada 3

```
5
```

Exemple de sortida 3

```
1 [5] [5]
```

Exemple d'entrada 4

```
9 7
```

Exemple de sortida 4

```
2 [7 9] [9 7]
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T21:33:01.940Z