Duplicació dels elements majors i eliminació dels elements menors que l'anterior d'una llista doblement encadenada, circular i amb fantasma X66861_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 *duplica_majors_elimina_menors*()

que duplica els elements majors que l'anterior i elimina els elements menors que l'anterior del p.i. El primer element i els iguals a l'anterior no es dupliquen ni s'eliminen.
Cal enviar a jutge.org només la implementació del mètode *duplica_majors_elimina_menors*.
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 duplica_majors_elimina_menors();
    // Pre: True
    // Post: S'han duplicat els elements majors que l'anterior i
    // s'han eliminat els elements menors que l'anterior del p.i.
    // El primer element i els iguals a l'anterior no es dupliquen ni s'eliminen.
    // Exemple: [2 5 3 4 1 1] => [2 5 5 4 4 1]
};
```

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 *duplica_majors_elimina_menors*.

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 duplicat els elements majors que l'anterior i eliminat els elements menors que l'anterior: 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 *duplica_majors_elimina_menors*. Segueix estrictament la definició de la classe de l'enunciat.

Exemple d'entrada 1

```
2 5 3 4 1 1
```

Exemple de sortida 1

```
6 [2 5 5 4 4 1] [1 4 4 5 5 2]
```

Exemple d'entrada 2

```
0 2 3 4 6 9
0 2 3 3 6 9
0 2 3 3 3 9
0 2 3 3 3 3
```

Exemple de sortida 2

```
11 [0 2 2 3 3 4 4 6 6 9 9] [9 9 6 6 4 4 3 3 2 2 0]
10 [0 2 2 3 3 3 6 6 9 9] [9 9 6 6 3 3 3 2 2 0]
9 [0 2 2 3 3 3 3 9 9] [9 9 3 3 3 3 2 2 0]
8 [0 2 2 3 3 3 3 3] [3 3 3 3 3 2 2 0]
```

Exemple d'entrada 3

```
9 8 7 4 3 1
9 8 7 7 3 1
9 8 7 7 7 1
9 8 7 7 7 7
```

Exemple de sortida 3

```
1 [9] [9]
2 [9 7] [7 9]
3 [9 7 7] [7 7 9]
4 [9 7 7 7] [7 7 7 9]
```

Exemple d'entrada 4

Exemple de sortida 4

```
0 [] []
```

Exemple d'entrada 5

```
3
-3
```

Exemple de sortida 5

```
1 [3] [3]
1 [-3] [-3]
```

Exemple d'entrada 6

```
0 1
0 -1
0 0
```

Exemple de sortida 6

```
3 [0 1 1] [1 1 0]
1 [0] [0]
2 [0 0] [0 0]
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T21:21:07.596Z